

32

JANVIER 1987

FORTH
APL
PASCAL

LES
RESEAUX

L'Europe, notion vaste et floue, réalité et utopie, représente avant tout pour nous, simples particuliers, une certaine idée de la liberté. Celle de recevoir et communiquer des informations entre habitants de différentes nations. Au delà des grands discours concernant les problèmes agricoles et industriels, il existe un tissu social d'où émergent des groupuscules ayant des intérêts communs. C'est le cas notamment des groupements d'utilisateurs du langage FORTH. Ces groupes existent partout en Europe, mais le seul à avoir répondu à notre appel est le FIG HAMBOURG (RFA).

Un groupement d'utilisateurs, c'est aussi la réunion de compétences permettant un dialogue plus actif et suivi. La barrière linguistique peut être abolie en traduisant les articles et programmes dans les langues respectives des différents groupements. Par ce moyen, nous pourrons accroître l'intérêt de notre magazine. L'expression multi-associative n'est pas une idée nouvelle, nous l'avons déjà tentée à plusieurs reprises (AUDIOCLUB des PTT, PPC-T, OUF, CIA, etc...), mais l'étendre au niveau européen reste une gageure, il faut la tenter.

Dans un tout autre registre, nous constatons la disparition de plusieurs revues dites 'verticales' dans le domaine de la micro-informatique et la mutation du contenu de certains mensuels. En fait, malgré l'apparition des nouveaux langages, BASIC continue à monopoliser les pages de ces magazines. Il n'en est pas de même dans les revues étrangères. Pour n'en citer que quelques-unes, le langage C s'impose de plus en plus dans BYTE et DrDOBB'S, PASCAL dans CHIP et DATA WELT ou COMPUTING TODAY. Or, l'information informatique, ce n'est pas seulement le chiffre de vente d'un tel ou le déficit d'un autre, ni les listings de transformation decimal/hexa ou jeux mièvres et longuets à taper, mais surtout les idées nouvelles, les théories et les expériences. Le rôle de JEDI est et restera la collection et la diffusion de la connaissance informatique fondamentale sous tous ses aspects. Si cet aspect devient Européen, il ne peut qu'être profitable pour tout le monde.

SOMMAIRE

ASSOCIATION JEDI 8, rue Poirier de Narçay 75014 PARIS
Tel: (1) 45.42.88.90 (de 10h à 18h)

FORTH

Dans ce propos, les diverses manières de développer une fonction en 83-Standard seront exposées avec pour exemple la création du mot INPUT.

Il existe de nombreuses manières de définir la fonction INPUT en langage FORTH, tout dépendant du niveau de sécurité souhaité et de la nature du résultat obtenu. La première manière, la plus simple, consiste à reproduire la ré-entrance de l'interpréteur externe de FORTH:

```
: DINPUT ( —— d)
QUERY 32 WORD NUMBER ;
```

De ce mot découle son homologue INPUT délivrant un nombre 16 bits:

```
: INPUT ( —— n)
DINPUT DROP ;
```

On pourrait se satisfaire de cette première approche. Mais tout programmeur sait que les utilisateurs ont une fâcheuse tendance à confondre les lettres O et les chiffres 0, quand ce n'est pas les doigts à coté des touches ou carrément l'activation de RETURN avant d'avoir entré un nombre. Or, nos deux premières définitions ne fonctionnent pas correctement si le nombre tapé n'est pas explicitement un nombre. De plus, si on entre un nombre 16 bits (sans point décimal), c'est un entier qui est déposé sur la pile de données. De même, si on entre un entier double précision, se sera un entier 32 bits qui sera déposé sur la pile.

Etant donné qu'il n'est pas question d'exiger de la part de l'utilisateur (qui lui n'en a cure de votre programme, et il a raison car il n'est qu'utilisateur...) de taper:

```
1234567.
```

quand il tapera instinctivement:

```
1234567
```

ce qui bien entendu ne rentre pas dans un format 16 bits. On va donc substituer le mot NUMBER? au mot NUMBER dans la définition de INPUT.

```
: INPUT ( —— d fl)
QUERY 32 WORD
NUMBER? ;
```

L'exécution de INPUT délivre le nombre double précision tapé en réponse à INPUT précédé d'un flag booléen. Ce flag booléen est vrai si la réponse est un nombre, elle est fausse dans les autres cas:

```
réponse à INPUT: 55     empile   55  0  -1
réponse à INPUT: 55.    empile   55  0  -1
réponse à INPUT: TRUC   empile   0   0   0
réponse à INPUT: <ret>  empile   0   0   0
```

Encore une fois, cette solution serait satisfaisante avec le petit arrangement suivant:

```
: INPUT ( —— d)
QUERY 32 WORD NUMBER? DROP ;
```

Mais on pourrait se servir utilement de ce petit drapeau booléen pour faire réexécuter l'entrée du nombre tant que celui-ci n'est pas valide:

```
: INPUT ( d ——)
BEGIN
   QUERY 32 WORD NUMBER? NOT
WHILE
   2DROP
REPEAT ;
```

Là c'est nettement plus sécurisé. Le petit malin qui prendra le clavier pour un défouloir à phalanges risque fort de rester bloqué avec des entrées numériques du style:

```
12ER4TRE56773JHLGJ67657676GHG76765
```

Mais à toute solution peut venir se greffer un problème. Si votre écran contient des informations, ce genre de manipulation risque de vous obliger à refaire le décor si le défoulement de l'individu appelé utilisateur vient empieter sur l'affichage dont la signification peut être vitale pour la suite des opérations. On prévoiera donc d'effacer et de faire revenir à la position de départ le curseur, ceci tant que l'information demandée n'est pas un nombre:

```
: INPUT ( —— d)
#OUT @
BEGIN
   DUP #OUT ! QUERY 32 WORD NUMBER NOT
WHILE
   2DROP DUP #OUT @ SWAP -
   DUP BACKSPACES DUP SPACES BACKSPACES
REPEAT
ROT DROP ;
```

Ca commence à s'étoffer, non? Bien entendu, votre utilisateur obstiné à confondre les 'Os' et les 'zerOs' ne court plus le risque de se répandre en essais successifs au travers de votre .... (tableur, traitement de texte, base de données: <- rayer la mention inutile). Mais par contre, rien ne l'empèche, histoire de voir et de vous embèter, d'essayer un nombre du genre:

```
12345676789012445678901234567890134567890913345667
```

Forth n'appréciera guère (j'ai moi-même planté un serveur visiblement pas au point avec ce truc, puis j'ai tapé EXP 'LIST', ce qui m'a donné le mode d'emploi, un peu à la manière de HELP avec dBASE...).

On peut envisager de limiter le nombre de caractères à taper:

```
: #INPUT ( u —— d)
#OUT @
BEGIN
   DUP #OUT ! TIB OVER EXPECT
   SPAN@ TIB ! BLK OFF >IN OFF 32 WORD
   NUMBER? NOT
WHILE
   2DROP DUP #OUT @ SWAP -
   DUP BACKSPACES DUP SPACES BACKSPACES
REPEAT
ROT DROP ROT DROP ;
```

Ne vous étonnez pas si le remplacement de QUERY par EXPECT nous oblige à rallonger la définition. Ce complément provient du contenu de la définition de QUERY et est listable par SEE QUERY.

Je vous encourage vivement à procéder à vos propres expériences. Bien entendu, n'hésitez pas à vous servir de DEBUG pour visualiser la trace de l'exécution de votre fonction INPUT.

Maintenant, notre fonction #INPUT est inviolable et sécurisée. Pour vous en convaincre, essayez:

```
DARK 10 10 AT .( !    !) 11 10 AT 4 #INPUT CR
```

Tout ce que vous taperez entre les deux points d'exclamation et qui ne sera pas constitué de quatres chiffres sera obstinément refusé.

Attention, la fonction AT est vectorisée et n'est peut-être pas initialisée correctement sur votre système.

Les définitions précédemment décrites sont spécifiques au FORTH 83-Standard et tournent sur les versions MSDOS et CP/M.

Cet exemple peut certainement être amélioré et adapté:

- lecture à un format spécifique, du genre INPUT-USING

```
" ==/==/==" INPUT-USING
```

De même, le nombre déposé sur la pile peut devenir un nombre 16 bits, flottant ou fractionnaire en fonction du contenu d'un pointeur paramétré par une application spécifique; pensez à la vectorisation... ■

S'il est un domaine difficile à appréhender, c'est bien celui de la télématique et des réseaux. Encore très mal exploité en France en dehors des applications à caractère commercial, il est pourtant promis à un avenir certain grâce à la chute vertigineuse des coûts des différents matériels et la croissance très rapide de la capacité des systèmes.

La télématique dont il sera question ici n'a rien à voir avec les messageries roses fleurissant sur T3 (3615). Notre télématique est celle de l'Europe. Elle permet l'accès à des documents et des ressources rares, un dialogue privilégié avec des concepteurs. En France, les précurseurs se nomment OUF et CALVADOS, sans oublier TRANSPAC. Mais pour celui qui maitrise plus d'une langue (l'anglais surtout!!), les contacts hors de nos frontières lui ouvrent des perspectives infinies.

Bien entendu, le crédit de votre compte bancaire doit être à la mesure de votre curiosité, sinon gare à la facture téléphonique. Mais l'avouerai-je, j'ai trouvé plus de renseignements en une heure de connexion sur ForthTREE (voir dans ce même numéro l'article consacré au FIG HAMBOURG) que je n'ai récolté de trucs et d'astuces vraiment utiles sur T3 en un an.

Notre seul souhait: permettre le développement des services télématiques thématiques et ouverts. OUF en est le meilleur exemple en France.

En attendant ces nouveaux réseaux, voici les numéros d'appel de quelques réseaux accessibles en Europe, accompagnés des spécifications techniques d'accès:

19 44 224 641 585	ABERDEEN	UK	V23	PRESTEL
19 44 903 420 13	ABBS WORTH	UK	V21	N 8 1
19 44 225 232 76	BABBS	UK	V21	N 8 1
19 44 394 276 306	BABBS 1	UK	V21	N 8 1
19 44 269 778 956	BABBS 2	UK	V21	N 8 1
19 44 742 667 983	BASUG	UK	V21	N 8 1
19 44 707 328 723	BBS CHILT	UK	V21	N 8 1
19 44 506 385 26	BBS LIVING	UK	V21	N 8 1
19 44 243 511 077	BBS SOUTHE	UK	V21	N 8 1
19 44 948 753 78	BETTISFIEL	UK	V21	N 8 1
19 49 104 930 1511	BILDSCHIRM	RFA	V23	PRESTEL
19 44 268 251 22	BITEC	UK	V21	N 8 1
19 44 827 288 810	BIRMING.N	UK	V21	N 8 1
19 44 258 544 94	BLANDFORD	UK	V21	N 8 1
19 44 295 720 812	BLOXAM	UK	V21	N 8 1
19 44 462 677 177	BULLETIN	UK	V23	PRESTEL
19 1 617 861 9774	BYTE	USA	BELL 103	N 8 1
19 44 699 321 4	CBBS CUMBR	UK	V21	N 8 1
19 44 139 921 36	CBBS LONDO	UK	V21	N 8 1
19 44 207 543 555	CBBS-NE	UK	V21	N 8 1
19 44 392 531 16	CBBS-SW	UK	V21	N 8 1
19 44 387 531 16	CBBS-SW	UK	V23	SCROLLING
19 44 486 225 174	CBBS Surey	UK	V21	N 8 1
19 44 707 328 723	CHILTERN	UK	V21	N 8 1
19 44 160 641 94	CITY BB	UK	V21	N 8 1
19 44 524 603 99	CNOL	UK	V21	N 8 1
19 44 163 130 76	COMPUTER A	UK	V21	N 8 1
19 44 874 711 147	COMUN HOPE	UK	V21	N 8 1
19 44 702 546 373	C-VIEW	UK	V23	PRESTEL
19 44 167 918 88	DISTEL	UK	V21	E 7 1
19 44 167 961 83	DISTEL	UK	AUTO BAUD RATE	
19 44 279 443 511	ESTELLE	UK	V21	E 8 1
19 44 279 441 188	ESTELLE	UK	V23	PRESTEL
19 44 279 441 222	ESTELLE	UK	V22	N 8 1
19 44 482 859 161	FORUM 80 H	UK	V21	N 8 1
19 44 926 398 71	FORUM 80 S	UK	V21	N 8 1
19 44 190 225 46	FORUM 80 W	UK	V21	N 8 1
19 44 482 497 150	HAMNET	UK	V21	N 8 1
19 44 506 385 26	LIVINGSTON	UK	V21	N 8 1
19 44 186 301 98	LONDON UG	UK	V21	N 8 1
19 44 384 635 336	MAILBOX-80 WARLEY	UK	V21	N 8 1
19 44 514 288 924	MAILBOX-80 LIVERPOOL	UK	V21	N 8 1
19 44 614 273 711	MANCHESTER	UK	V21	N 8 1
19 44 614 271 596	MANCHES. D	UK	V21	N 8 1
16 29 892 333	MANEL BBS	FRA	V21	E 7 1
19 44 702 552 941	MAPTEL	UK	V21	N 8 1
19 44 164 026 17	MBBS	UK	V21	N 8 1
19 44 157 922 88	MICRO LIVE	UK	V21	N 8 1
19 44 614 564 157	MICROWEB	UK	V21	N 8 1
087 883 434	MICRO-NET2	B	V21	N 8 1
041 796 666	MICRO-NET2	B	V21	N 8 1
010 228 099	MNWAVRE	B	V21	N 8 1
19 44 617 368 449	MOBBS	UK	V21	N 8 1
19 44 692 630 610	NBBS	UK	V21	N 8 1
19 44 692 630 186	NBBS C	UK	V21	N 8 1
19 44 795 842 324	NKABBS KEN	UK	V21	N 8 1
19 44 827 288 810	NOTHINGHAM	UK	V21	N 8 1
19 44 614 271 596	OBBS	UK	V21	N 8 1
19 49 6154 51 433	OBER-RAMST	RFA	V21	N 8 1
16 90 77 61 36	PICONET	FRA	V21	N 8 1
19 44 742 667 983	PIP	UK	V21	N 8 1
19 44 227 236 28	REWTEL	UK	V21	N 8 1
19 44 243 511 077	SOUTHERN B	UK	V21	N 8 1
19 44 126 890 014	SOUTHWEST	UK	V21	N 8 1
19 44 782 265 078	SITEC	UK	V21	N 8 1
19 44 782 265 078	STOKE ITEC	UK	V21	N 8 1
19 44 134 894 19	TBBS LONDO	UK	V21	N 8 1
19 44 134 178 40	TBBS METRO	UK	V21	N 8 1
19 44 602 289 783	TBBS NOTHI	UK	V21	N 8 1
19 44 258 544 94	TBBS BLAND	UK	V21	N 8 1
19 44 703 437 219	TBBS SOUTH	UK	V21	N 8 1
041 529 930	ULG	B	V21	E 7 1
19 44 903 420 13	WABBS	UK	V21	N 8 1
19 44 384 635 336	W MIDLANDS	UK	V21	N 8 1

Et pour nos adhérents habitant la région Est, ils pourront avantageusement se connecter sur les réseaux fonctionnant en Suisse:

19-4139 412505	Micronet Lausanne	F	24h
19-411 7803290	Excom Waedenswil	D	24h
19-4153 45458	PIM Schaffouse	D	24h
19-4137 231689	Norasia Fribourg	D/F/E	24h
19-4131 588939	Paddle Box Bern	D	24h
19-4131 360143	Vogelfutter Bern	D	24h
19-4162 519751	DIS Zofingen	D	24
19-4131 962106	Interdiscount	D	24
19-4157 461858	Data64 Argovie	D	22-09h
19-4161 388347	BMS Bale	D	22-..h
19-411 3122267	ZEV Zurich	D	19-09h +DI
19-4161 736639	CCC CH Bale	D	?
19-4161 509355	ECM Bale	D	22-..h
19-4171 411885	Paus-Box Rorschach	D	19-07h

Et pour la France? Les initiatives durables sont encore très ponctuelles. Cependant, nous disposons d'un atout non négligeable, le réseau TRANSPAC. Bien entendu, vous connaissez les fameux 3613, 3614 et 3615 (T1, T2 et T3). Mais saviez-vous que vous pouvez également vous connecter en 300 bauds et 1200 bauds full duplex. Ces deux points d'accès sont essentiellement réservés aux utilisateurs professionnels. C'est sur ces deux réseaux que l'on peut avoir accès à CALVADOS. ACCES :

300/300 Bauds	Tel. 36 01 91 00	Norme V21-mode Appel
1200/1200 Bauds	Tel. 36 00 91 22	Norme V22B-mode 2

TRANSPAC-UTILISATION DU PAD

Résumé des principales caractéristiques d'utilisation du réseau TRANSPAC à travers un PAD, extraites des Spécifications Techniques d'Utilisation du Réseau (S.T.U.R. du 05.07.84) par Arnaud DELMAS.

ETABLISSEMENT DE LA COMMUNICATION :

- Connexion du modem.
- Message "TRANSPAC".
- Faire le numéro (9 chiffres) du serveur, puis CR.
- Message "COM" si la liaison est établie.

NOTA:

1/ Avant de faire le numéro du serveur on peut envoyer des commandes au PAD sans préfixe.

2/ Si la liaison ne peut être établie, le PAD transmet un message d'erreur.

3/ Le PAD attend un numéro ou une commande pendant 60 sec ou 4 tentatives avant de déconnecter la ligne.

TRANSFERT DE DONNEES :

- Format : mots de 8 bits, pas de parité, 1 bit de stop
- Caractère de contrôle du PAD: DLE (^P)
- Contrôle de débit par le PAD: XOFF-DC3 (^S), XON-DC1 (^Q)

- Contrôle de débit par l'utilisateur: XOFF-DLE, XON-CR
- Transmission de DLE: DLE, DLE
- Taille max d'un paquet : 128 octs stand (32, 64, 256)
- Envoi d'un paquet: par caractère d'envoi ou délai max

NOTA:
Certaines caractéristiques peuvent etre modifiées, en changeant les paramètres du PAD. En particulier, il est possible de rendre le PAD transparent au caractère DLE, et de supprimer le contrôle du débit par le PAD.

COMMANDES DU PAD :

Format: <préfixe><commande>[+<commande>...]CR
Prefixe: DLE normalement, BREAK dans certain cas (le préfixe n'est utilisé que si la communication est établie)

- LIB Fin de communication
- RESET Envoi d'un paquet d'initialisation
- INT Envoi d'un paquet d'interruption
- STAT Etat de la communication (FREE/ENGAGED)
- PAR? Etat des paramètres du PAD
- PAR? <paramètre>[,...] Etat d'un ou plusieurs paramètres du PAD
- SET <paramètre>:<valeur>[,...] Modification d'un ou plusieurs paramètres du PAD
- SET? <paramètre>:<valeur>[,...] Modification puis relecture d'un ou plusieurs paramètres du PAD
- PROF <profil> Sélection d'un profil

PARAMETRES DU PAD :

1: Transparence à DLE
  1= DLE utilisé par le PAD
  0= PAD transparent pour DLE

2: Echo par le PAD
  1= Echo des caractères reçus par le PAD
  0= Pas d'écho

3: Caractère d'envoi de données
  126= Car. de contrôles + DEL
  0= Pas de car. d'envoi
  2= CR uniquement

4: Délai max d'envoi de données
  0= Pas de delai max
  1 à 255= Délai en 1/20e de Sec

5: Contrôle de débit par le PAD
  1= Xon-Xoff
  0= Pas de contrôle

6: Messages du PAD vers l'utilisateur
  1= Oui
  0= Messages non transmis

7: Effet d'un BREAK
  2= Envoi d'un paquet de réinitialisation
  0= Envoi d'un paquet de données
  1= Envoi d'un paquet d'interruption
  8= Préfixe de commande (identique a DLE)
  21= Envoi d'une procédure de BREAK

8: Arrêt de transmission des données
  0= Transmission
  1= Arrêt (données perdues)

9: Caractères nuls
  0= Non utilisés
  1 a 255= Nb de nulls après CR

10: Pliage de ligne
  0= Non utilisé
  1 à 255= Longueur max de ligne

11: Vitesse de transmission (non modifiable)

La première valeur de chaque paramètre est la valeur par défaut.

PROFILS:

0: Profil simple (valeurs par défaut) selon Avis X28

1 à 15: Profils prédéfinis

1: 0,0,0,20,0,0,2,0,0,0	profil transparent selon Avis X28
3: 1,0,2,80,0,1,21,0,0,0	utilisable avec KERMIT
4: 1,0,2,40,0,1,21,0,4,0	idem
6: 1,1,126,0,1,1,21,0,0,0	idem 0, mais traitement BREAK spécial

MESSAGES DU PAD:

- COM Communication établie
- LIB CONF Communication interrompue par l'utilisateur
- LIB XXX Communication interrompue pour la raison XXX
  - OCC Occupé
  - INV Demande de facilité invalide
  - NC Incident réseau
  - DER Dérangement
  - NA Non autorisé
  - NP Inconnu
  - RPE Erreur de procédure distante
  - ERR Erreur de procédure locale
  - PCV Refus de taxation au demandé
  - PAD Par le PAD
  - DTE nnn Par le serveur pour raison nnn
- RESET XXX Réinitialisation pour la raison XXX
- ERROR Erreur de syntaxe de commande du PAD

TRANSMISSION DE FICHIERS:

La transmission par KERMIT est la meilleure méthode car elle n'interfère pas avec le caractère DLE ni les caractères de contrôle utilisés par le serveur. Pour la configuration de KERMIT il faut tenir compte des points suivant:

- Caractère de fin de paquet: CR
- Protocole Xon-Xoff inutile
- Pas de problème de longueur de paquets
- Transmission de fichiers binaires sans encodage du 8ème bit

Les paramètres du PAD seront programmés ainsi:

1:1 DLE non utilisé par KERMIT
2:0 Suppression d'écho
3:2 Envoi des données sur car. de fin de bloc
4:10 Temporisation 500 ms pour caractères hors paquet
5:0 Pas de contrôle de débit par le PAD
6:1 Sans importance
7:2 Sans importance (dépend du Serveur)
8:0
9:0 Pas de nulls, en principe
10:0 Pas de pliage de ligne

La transmission par XMODEM est en principe possible mais délicate, car elle nécessite la mise en transparence totale:

1:0 Pas de filtrage de DLE
2:0 Suppression d'écho
3:0 Pas de caractères d'envoi
4:10 Temporisation d'envoi 500 ms
5:0 Pas de contrôle de débit par le PAD
6:1 Sans importance
7:8 BREAK remplace le caractère DLE
8:0
9:0 Pas de nulls après CR
10:0 Pas de pliage de ligne

COMMANDES SPECIFIQUES A LA NORME VIDEOTEX

Si vous disposez d'un Minitel, son utilisation en tant que serveur monovoie par retournement du modem est facile à mettre en oeuvre. De même, pour les nouveaux matériels type TELESTRAT, THOMSON TO9+ et IBM avec carte CORTEX (V21,V23), l'émulation de toutes les fonctions décodables par un Minitel sont contrôlées par programme. Pour ceux dont les projets télématiques sont ambitieux, voici la liste complète des fonctions TELETEL:

ESC 9 g	(1B 39 67)	Déconnexion
ESC 9 h	(1B 39 68)	Connexion
ESC ; ½ <rcpt> <emtt>	(1B 3B 60)	Blocage module
ESC ; a <rcpt> <emtt>	(1B 3B 61)	Aiguillage module
ESC : d <rcpt>	(1B 3A 64)	Diffusion restreinte

ESC : e <rcpt> (1B 3A 65) Diffusion systématique

ESC : b <modu> (1B 3A 62) Demande status module
ESC ; c <modu> <code> (1B 3B 62) Réponse status module

avec

<modu>	<rcpt>	<emtt>
Ecran	X (58)	P (50)
Clavier	Y (59)	Q (51)
Modem	Z (5A)	R (52)
Prise	[(5B)	S (53)
Téléph.	ç (5C)	T (54)
Logic.	] (5D)	U (55)

avec <code>

bit 0	écran
bit 1	clavier
bit 2	modem
bit 3	prise
bit 4	module téléphonique
bit 5	logiciel spécifique
bit 6	toujours 1

ESC : i <code> (1B 3A 69) Start mode fonctionnement
ESC : j <code> (1B 3A 6A) Stop mode fonctionnement

avec <code>=

B (42)	80 colonnes (seulement avec nouveau modèle type MATRA)
C	rouleau
D	procédure corr. erreur
E	mode enseignement
F	loupe haut
G	loupe bas

ESC 9 l (1B 39 6C) Demande retournement modem (E:1200 R:75)
ESC 9 m (1B 39 6D) Demande retournement modem (E:75 R:1200)

ESC 9 p (1B 39 70) Demande status terminal
ESC : q <code> (1B 3A 71) Réponse status terminal

avec <code>=

bit 0	
bit 1	1 (E:75 R:1200)
bit 2	module téléphonique
bit 3	connecté
bit 4	péripherique
bit 5	module logiciel

ESC 9 r (1B 39 72) Demande status fonctionnement
ESC : s <code> (1B 3A 73) Réponse status fonctionnement

avec <code>=

bit 0	1 si 80 colonnes
bit 1	1 si rouleau
bit 2	1 si correction erreur
bit 3	1 si mode enseignement
bit 4	loupe haute
bit 5	loupe basse

ESC 9 t (1B 39 74) Demande status vitesse
ESC : u <code> (1B 3A 75) Réponse status vitesse
ESC : k <code> (1B 3A 6B) Programmation vitesse

avec <code>=

bit 0	1 si réception 75
bit 1	1 si réception 300
bit 2	1 si réception 1200
bit 3	1 si émission 75
bit 4	1 si émission 300
bit 5	1 si émission 1200
bit 6	toujours à 1

ESC 9 v (1B 39 76) Demande status protocole
ESC : w <code> (1B 3A 77) Réponse status protocole

avec <code>=

bit 0	les acquittements -> modem
bit 1	les acquittements -> prise

ESC : f <nb> (1B 3A 66) mode transparence
les <nb> octets suivants ne sont pas interprétés
le protocole acquitte avec ^S W (13 57)

REPONSES à changement

^S A	(13 41)	touche ENVOI
^S B	(13 42)	touche RETOUR
^S C	(13 43)	touche REPETITION
^S D	(13 44)	touche GUIDE
^S E	(13 45)	touche ANNULATION
^S F	(13 46)	touche SOMMAIRE
^S G	(13 47)	touche CORRECTION
^S H	(13 48)	touche SUITE
^S I	(13 49)	touche CONNECT/FIN

^S J	(13 4A)	demande de procédure correc. erreur
^S K	(13 4B)	demande d'arrèt de cette procédure
^S L	(13 4C)	demande retournement (E:1200 R:75)
^S M	(13 4D)	demande retournement (E:75 R:1200)
^S Q	(13 51)	vitesse modem
^S R	(13 52)	module téléphonique
^S S	(13 53)	porteuse (connection)
^S T	(13 54)	périphérique
^S U	(13 55)	modules logiciels
^S V	(13 56)	mode de fonctionnement demandé par ligne
^S W	(13 57)	acquittement mise en transparence

ENVOI	^S A	(13 41)	
REPET.	^S C	(13 43)	
GUIDE	^S D	(13 44)	
ANNUL.	^S E	(13 45)	
SOMM.	^S F	(13 46)	
CORREC.	^S G	(13 47)	
SUITE	^S H	(13 48)	
ESPACE		(20)	
LOUPE			local
CON-FIN	^S I	(13 49)	vers modem

lettres spéciales 3 codes ^V signe lettre (16 SS LL)
ou ^V signe

£	(23)	livre
$	(24)	dollar
#	(26)	dièze
,	(2C)	flèche gauche
-	(2D)	flèche haut
.	(2E)	flèche droite
/	(2F)	flèche bas
0	(30)	degré
1	(31)	+ ou -
8	(38)	divisé
<	(3C)	1/4
=	(3D)	1/2
>	(3E)	3/4
A	(41)	½
B	(42)	'
C	(43)	circonflèxe
H	(48)	tréma
K	(4B)	cédille
j	(6A)	OE liés
z	(7A)	oe liés

^N (OE) passe en semi graphique entre (20) et (7E)
^O (OF) retour en normal

adressage curseur
^_ m n (1F)m n
ligne mn col1 pour m et n entre 0 et 1 (30) et (39)
^_ l c (1F)l c
ligne 1-40 col c-40 l et c en binaire à partir de A

Ex: pour ligne 6 col 1 faire
ou ^_ 0 6 (1F 30 36)
ou ^_ F A (1F 46 41)

^^ (1E) retour début ligne 1 (home)
^L (OC) efface écran et retour ligne même colonne
^X (18) efface fin de ligne avec le fond défini

pour sortir de ligne status <LF> ou positionnement

Attributs: <ESC> code

Caractère		Fond	
@ (40)		P (50)	noir
A (41)		Q (51)	rouge
B (42)		R (52)	vert
C (43)		S (53)	jaune
D (44)		T (54)	bleu
E (45)		U (55)	magenta
F (46)		V (56)	cyan
G (47)		W (57)	blanc
H (48)	clignote		
I (49)	fixe		
J (4A)	fin incrust.		
K (4B)	début incrust. (en transmission radio: ANTIOPE)		
L (4C)	taille norm.		
O (4F)	taille double		
M (4D)	hauteur double		
N (4E)	largeur double		
X (58)	masquage		
(5F)	démasquage		
Y (59)	fin soulign.		

suite en page 6

Nous avons vu précédemment comment calculer en APL les racines d'une équation du 2ème degré par une suite d'opérations simples dont voici le rappel:

1) Saisie de données:

```
A ← ⎕
B ← ⎕
C ← ⎕
```

2) Calcul du déterminant:

```
Δ ← (B * 2) - (4 × A × C)
```

Là, deux variantes possible:

a) si   Δ ≥ 0

on calcule:

```
D ← Δ * 0.5
```

3) Calcul des racines réelles:

1ère racine: $R1 \leftarrow ((-B) + D) \div (2 \times A)$
2ème racine: $R2 \leftarrow ((-B) - D) \div (2 \times A)$

b) si   Δ < 0

on calcule:

```
D ← (-Δ) * 0.5
```

4) Calcul des racines complexes:

partie réelle: $R \leftarrow (-B) \div (2 \times A)$
partie immaginaire: $I \leftarrow D \div (2 \times A)$

Nous pouvons grouper une séquence d'opérations en une entité appelée "fonction". Il suffit ensuite d'appeler la fonction par son nom pour que toute la séquence se déroule.

Nous pourrons définir une première fonction que nous appelerons SAISIE permettant de fixer les valeurs des paramètres A, B et C. Sans préciser ici les détails de la syntaxe du langage, nous dirons seulement que cette fonction peut s'écrire:

```
[0]    SAISIE
[1]    ' A = '
[2]    A←⎕
[3]    ' B = '
[4]    B←⎕
[5]    ' C = '
[6]    C←⎕
```

Ensuite, nous pouvons écrire une fonction DETERMINANT calculant la valeur du déterminant Δ ainsi que celle du paramètre D, égale à la racine carrée de la valeur absolue de Δ.

Cette fonction pourra même faire appel à la première pour la saisie des données. Nous écrirons:

```
[0]   DETERMINANT
[1]   SAISIE
[2]   Δ←(B*2)-(4×A×C)
[3]   D←(|Δ)*0.5
```

Ensuite, selon le signe de Δ, nous appelerons soit la fonction REELLES qui calculera et affichera les racines réelles:

```
[0]   REELLES
[1]   'R1 = '
[2]   ((-B)+D)÷(2×A)
[3]   'R2 = '
[4]   ((-B)-D)÷(2×A)
```

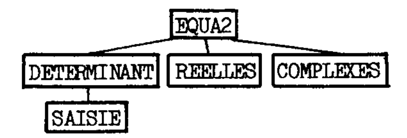
Soit la fonction COMPLEXES qui calculera et affichera la partie réelle, puis la valeur absolue de la partie imaginaire des deux racines conjuguées.

```
[0]   COMPLEXES
[1]   'R = '
[2]   (-B)÷(2×A)
[3]   'I = '
[4]   D÷(2×A)
```

On pourra "ficeler" le tout par une fonction appelée EQUA2, laquelle fera appel successivement à la fonction DETERMINANT laquelle fait elle-même appel à SAISIE pour avoir les valeurs des paramètres, puis, selon le cas, à la fonction REELLES ou à la fonction COMPLEXES.

```
[0]   EQUA2
[1]   DETERMINANT
[2]   →((Δ≥0),(Δ<0))/RE,CX
[3]   RE:REELLES
[4]   →FIN
[5]   CX:COMPLEXES
[6]   FIN:
```

Nous disposons maintenant pour calculer les racines d'une équation du 2ème degré d'un "progiciel" comprenant cinq fonctions, et structuré de la manière suivante:



Ca n'est sans doute pas la manière la plus simple et la plus économique de calculer les racines d'une équation du 2ème degré, mais ce petit exemple permet de comprendre (du moins nous l'espérons) le mécanisme des fonctions du langage APL.

F.ESPINASSE

suite de la page 5

Z (5A) début soulignement

ç (5C) fond normal
] (5D) fond inverse
^ (5E) fond transpar.

répétition ^Rn (12)n répète n-40 fois le car' précédent
Ex A ^R ^J (41 12 4A) répète 10 A derrière le A (total 11)

<del> (7F) écrit un caractère pavé couleur caractère

demande position
<esc> a   (1B 61) répond avec code adressage

^E   (05)   demande identification

^Q   (11)   curseur visible
^T   (14)   curseur invisible

<esc> £ <sp> attribut   (1B 23 20 at)
attribut sur tout l'écran

<esc> £ ! attribut   (1B 23 21 at)
attribut sur toute la ligne

<esc> £ <sp> _   (1B 23 20 5F)
révèle parties masquées

code mal définis
<esc> action param   (1B 3x 4p) commande d'un periph.

5	(35)	mise en route
6	(36)	arrêt
7	(37)	attente
à	(40)	recopie papier
A	(41)	enregistrement
B	(42)	roll up autorisé
C	(43)	roll down autorisé

essayer n guide   et qqch guide   pour taxes



COMPRESSEUR DE FICHIERS EN CODAGE DE HUFFMAN *par Lindo ZHANG*

```
PROGRAM CMPRSS;
{ Compression de fichiers }
(*$R-*)
(*$A+*)

CONST
  N0=257;
  NM0=256;
  LRECORD=10;     (* LONGEUR D'UN ENREGISTREMENT "OBJTYPE" *)
  MAXF=1E30;      (* BORNE SUP. DES FREQUENCES *)


TYPE

  CODE=STRING[40];

  POINTER=^NODETYPE;

  OBJTYPE=RECORD
            FREQUENCE   : REAL;      (* FREQUENCE *)
            ASCII       : INTEGER;   (* CODE ASCII =256 ==> EOF*)
            NODE        : POINTER;   (* POINTE SUR LE NOEU DANS L'ARBRE S'IL *)
          END;                       (* EST UN SOUS-ARBRE ET =NIL S'IL EST   *)
                                     (* UN CARACTERE                         *)
  NODETYPE=RECORD
             LEFTCHILD  : POINTER;   (* POINTE LES 2 NOEUX GAUCHES *)
             LEFTCHAR   : INTEGER;   (* CARACTERE GAUCHE           *)
             RIGHTCHILD : POINTER;
             RIGHTCHAR  : INTEGER;
           END;

  OBJTYPE1=ARRAY [0..N0] OF OBJTYPE; (* 258 objets *)

LABEL PROMPT;

VAR
  CRC,CRC0     : INTEGER;    (* Cheksum *)
  OBJET        : OBJTYPE1;   (* Table de frequences *)
  OBJET1       : OBJTYPE1;   (* Sauvegarde de la table *)
  CODAGE       : ARRAY[0..NM0,0..9] OF BYTE ABSOLUTE OBJET;
  PROOT        : ^NODETYPE;  (* RACINE DE L'ARBRE *)
  I,N          : INTEGER;
  C            : CODE;       (* Sert a l'impression des codes *)
  LINECOUNT    : BYTE;       (* Compteur des lignes imprimees *)
  PRTFLAG      : BOOLEAN;
  LEN0,LEN     : REAL;       (* Longeur du fich. orginal et celle *)
                             (* du fich. compresse *)
  NOM0,NOM1    : STRING[16]; (* Noms de fichiers    *)
  CMDLIN       : STRING[21];
  FICH0,FICH1  : FILE;
  BUFFIN       : ARRAY [0..130] OF BYTE;
  PTFICH0      : BYTE;
  PTFICH1      : INTEGER;
  TAILTAMPON   : INTEGER;    (* LONGUEUR TAMPON EN NBR DE BYTES *)
  TAMPON       : ^BYTE;
  LENTAMPON    : INTEGER;    (* LONGUEUR TAMPON EN NBR D'ENREGISTREMENT *)
  STACK        : BYTE;       (* <3 lorsqu'on pousse des bytes dans le buffet *)
  TABLE        : INTEGER;    (* POSITION DE LA TABLE DANS LE FICH. EN SORTIE *)
  TEMP         : REAL;
  SAVEBITS     : BYTE;
  PTBITS       : BYTE;
  CH           : BYTE;
  CARACT       : CHAR;
  UNTILFLAG    : BOOLEAN;


{--------------------------------------------------------------------}
PROCEDURE OPENIN;  { OUVERTURE DU FICHIER EN ENTREE }
BEGIN
  ASSIGN(FICH0,NOM0);
  {$I-}
  RESET(FICH0);
  {$I+}
  PTFICH0:=131;
  STACK:=3;
  CRC:=0
END;


{--------------------------------------------------------------------}
FUNCTION POPCHAR(VAR CH: BYTE): BOOLEAN;
{Lit un car. depuis le buffet
 Retourne FALSE si l'on est a la fin du fichier }
LABEL  FIN;
VAR    I  :BYTE;
```

```
BEGIN
  IF PTFICH0>130 THEN
  BEGIN
    IF EOF(FICH0) THEN BEGIN POPCHAR:=FALSE;GOTO FIN END;
    BLOCKREAD(FICH0,BUFFIN[3],1);
    FOR I:=3 TO 130 DO CRC:=CRC+BUFFIN[I];
    PTFICH0:=3
  END;
  POPCHAR:=TRUE;
  CH:=BUFFIN[PTFICH0];
  PTFICH0:=PTFICH0+1;
FIN:END;


{-----------------------------------------------------------------------}
PROCEDURE PUSHCHAR(CH : BYTE);
{ Repousser un octet dans le buffet }
BEGIN
  PTFICH0:=PTFICH0-1;
  BUFFIN[PTFICH0]:=CH
END;


{-----------------------------------------------------------------------}
FUNCTION GETCHAR(VAR CH: BYTE): BOOLEAN;
{ Ce Programme remplace une sequence de bytes identiques par
 $90 suivi du nombre de repetitions (maxi. 255) }
VAR CH1    : BYTE;
    COUNT  : BYTE;
    NOFIN  : BOOLEAN;
BEGIN
  NOFIN:=POPCHAR(CH);
  GETCHAR:=NOFIN;
  IF NOFIN THEN  { Pas la fin du fichier }
  IF STACK<=2 THEN STACK:=SUCC(STACK) ELSE
    IF CH=$90 THEN
    BEGIN
      IF POPCHAR(CH1) THEN
      BEGIN
        IF CH1<>CH THEN
        BEGIN
          PUSHCHAR(CH1);
          PUSHCHAR(0);
          STACK:=2
        END ELSE
        BEGIN
          COUNT:=1;
          NOFIN:=TRUE;
          WHILE (CH1=CH) AND (COUNT<255) AND NOFIN DO
          BEGIN
            COUNT:=SUCC(COUNT);
            NOFIN:=POPCHAR(CH1)
          END;
          IF NOFIN THEN PUSHCHAR(CH1);
          STACK:=0;
          PUSHCHAR(COUNT);
          PUSHCHAR($90);
          PUSHCHAR(0)
        END
      END ELSE
      BEGIN
        PUSHCHAR(0);
        STACK:=2
      END
    END ELSE
        IF POPCHAR(CH1) THEN
          IF CH1<>CH THEN PUSHCHAR(CH1) ELSE
             IF POPCHAR(CH1) THEN
               IF CH1<>CH THEN
               BEGIN
                 PUSHCHAR(CH1);
                 PUSHCHAR(CH);
                 STACK:=2
               END ELSE
               BEGIN
                 COUNT:=2;
                 NOFIN:=TRUE;
                 WHILE (CH1=CH) AND (COUNT<255) AND NOFIN DO
                 BEGIN
                   COUNT:=SUCC(COUNT);
                   NOFIN:=POPCHAR(CH1)
                 END;
                 IF NOFIN THEN PUSHCHAR(CH1);
                 STACK:=1;
                 PUSHCHAR(COUNT);
                 PUSHCHAR($90)
               END;
    CH:=CH XOR -1
  END;
```

```
{----------------------------------------------------------------------}

PROCEDURE GETFREQ(VAR N: INTEGER);
{ Generation de la table de frequences }
{ Retourne dnas N le nbr. d'objets }
VAR
  I,J     : INTEGER;
  ENCORE  : BOOLEAN;
  CH      : BYTE;
BEGIN
  FOR I:=0 TO NM0 DO
    WITH OBJET[I] DO
    BEGIN
      FREQUENCE:=0;
      ASCII:=I;
      NODE:=NIL
    END;
  ENCORE:=GETCHAR(CH);
  WHILE ENCORE DO
  BEGIN
    OBJET[CH].FREQUENCE:=OBJET[CH].FREQUENCE+1;
    ENCORE:=GETCHAR(CH)
  END;
  OBJET[NM0].FREQUENCE:=1;
  OBJET[N0].FREQUENCE:=MAXF;
  MOVE (OBJET,OBJET1,N0*LRECORD); { SAUVER LA TABLE DE FREQUENCES }
  {Skip les zero }
  I:=0;
  WHILE (OBJET[I].FREQUENCE<>0) DO I:=SUCC(I);
  IF I>=NM0 THEN N:=N0 ELSE
  BEGIN
    J:=I;
    WHILE (I<=NM0) DO
    BEGIN
      I:=SUCC(I);
      IF OBJET[I].FREQUENCE<>0 THEN
      BEGIN
        OBJET[J]:=OBJET[I];
        J:=SUCC(J)
      END
    END;
    N:=PRED(J)
  END
END;

{----------------------------------------------------------------------}

{$A-}
PROCEDURE TREE(VAR ROOT:NODETYPE;N:INTEGER);
{ Generation de l'arbre binaire du codage }
LABEL CONT;
VAR
  P0     : POINTER;
  I,J,K  : INTEGER;
  F      : REAL;
BEGIN
  IF N<=2 THEN WITH ROOT DO
     BEGIN
       LEFTCHILD:=OBJET[0].NODE;
       LEFTCHAR:=OBJET[0].ASCII;
       RIGHTCHILD:=OBJET[1].NODE;
       RIGHTCHAR:=OBJET[1].ASCII;
     END ELSE
     BEGIN
       NEW(P0);
       P0^.LEFTCHILD:=OBJET[0].NODE;
       P0^.RIGHTCHILD:=OBJET[1].NODE;
       P0^.LEFTCHAR:=OBJET[0].ASCII;
       P0^.RIGHTCHAR:=OBJET[1].ASCII;
       F:=OBJET[0].FREQUENCE+OBJET[1].FREQUENCE;
       J:=2;
       K:=N;
       WHILE (J<PRED(K)) DO
       BEGIN
         I:=(J+K) SHR 1;
         IF F>OBJET[I].FREQUENCE THEN J:=I ELSE K:=I;
       END;
       I:=K;
CONT:  IF I>2 THEN MOVE(OBJET[2],OBJET[0],(I-2)*LRECORD);
       IF I<N THEN MOVE(OBJET[I],OBJET[I-1],(N-I)*LRECORD);
       WITH OBJET[I-2] DO
         BEGIN
           FREQUENCE:=F;
           NODE:=P0
         END;
       OBJET[N-1].FREQUENCE:=MAXF;
       TREE(ROOT,N-1)
     END
END;
```

```
{---------------------------------------------------------------------------}
{$A+}
PROCEDURE OPENOUT;
BEGIN
  ASSIGN(FICH1,NOM1);
  REWRITE(FICH1);
  PTFICH1:=1;
  TAILTAMPON:=MEMAVAIL;
  IF (TAILTAMPON<1152) AND (TAILTAMPON>=0) THEN
  BEGIN
    WRITELN(^A^D'Memoire insuffisante');
    HALT
  END;
  TAILTAMPON:=1152;
  LENTAMPON:=TAILTAMPON DIV 128;
  GETMEM(TAMPON,TAILTAMPON);
END;

{---------------------------------------------------------------------------}

PROCEDURE PUTCHAR(VAR TAMPON;CH: BYTE);
VAR
  BUFFOUT : ARRAY [1..1] OF BYTE ABSOLUTE TAMPON;
BEGIN
  IF PTFICH1>TAILTAMPON THEN
  BEGIN
    BLOCKWRITE(FICH1,BUFFOUT,LENTAMPON);
    PTFICH1:=1
  END;
  BUFFOUT[PTFICH1]:=CH;
  PTFICH1:=SUCC(PTFICH1)
END;
{---------------------------------------------------------------------------}

PROCEDURE PUTBITS(VAR CHAINE1;NBRBITS: BYTE);
VAR
  CHAINE     : ARRAY [1..1] OF BYTE ABSOLUTE CHAINE1;
  I,J,K      : BYTE;
BEGIN
  J:=1; K:=1;
  FOR I:=1 TO NBRBITS DO
  BEGIN
    IF (CHAINE[J] AND K)<>0 THEN SAVEBITS:=SAVEBITS OR PTBITS;
    IF PTBITS=128 THEN
    BEGIN
      PUTCHAR(TAMPON^,SAVEBITS);
      SAVEBITS:=0;
      PTBITS:=1
    END ELSE PTBITS:=PTBITS SHL 1;
    IF K=128 THEN
    BEGIN
      K:=1;
      J:=SUCC(J)
    END ELSE K:=K SHL 1
  END
END;
{---------------------------------------------------------------------------}

PROCEDURE CLOSEOUT(VAR TAMPON);
VAR REST    : BYTE;
    BUFFOUT : ARRAY [1..1] OF BYTE ABSOLUTE TAMPON;
BEGIN
  IF PTBITS<>1 THEN PUTCHAR(BUFFOUT,SAVEBITS);
  PTFICH1:=PRED(PTFICH1);
  REST:=PTFICH1 MOD 128;
  IF REST=0 THEN BLOCKWRITE(FICH1,BUFFOUT,PTFICH1 DIV 128) ELSE
  BEGIN
    FILLCHAR(BUFFOUT[PTFICH1+1],128-REST,$1A);
    BLOCKWRITE(FICH1,BUFFOUT,(PTFICH1 DIV 128)+1)
  END;
  CLOSE(FICH1)
END;
{---------------------------------------------------------------------------}
PROCEDURE PRTCODE(CH:INTEGER;F:REAL;C:CODE);
VAR
  LONG,I,J,K,CC : BYTE;
BEGIN
  IF PRTFLAG THEN
  BEGIN
    LINECOUNT:=LINECOUNT+1;
    IF CH=256 THEN WRITELN('EOF    ',F:10:0,'          ',C) ELSE
    BEGIN
      CC:=(CH XOR -1) AND 255;
      I:=CC DIV 16;
```

```
      IF I<10 THEN WRITE(CHR($30+I)) ELSE WRITE(CHR($41+I-10));
      I:=CC MOD 16;
      IF I<10 THEN WRITE(CHR($30+I)) ELSE WRITE(CHR($41+I-10));
      IF CC>=32 THEN WRITE('   ',CHR(CC)) ELSE WRITE('   .');
      WRITELN('  ',F:10:0,'          ',C)
    END
  END;
  LONG:=LENGTH(C);
  LEN:=LEN+LONG*(F/8);
  CODAGE[CH,0]:=LONG;
  J:=1;
  K:=1;
  CC:=0;
  FOR I:=1 TO LONG DO
  BEGIN
    IF C[I]='1' THEN CC:=CC OR J;
    IF J=128 THEN
    BEGIN
      CODAGE[CH,K]:=CC;
      CC:=0;
      K:=SUCC(K);
      J:=1
    END ELSE J:=J SHL 1
  END;
  CODAGE[CH,K]:=CC;
  IF (LINECOUNT>=20) AND PRTFLAG THEN
  BEGIN
    WRITELN;
    WRITE('<Une touche>');
    REPEAT UNTIL KEYPRESSED;
    LINECOUNT:=0;
    CLRSCR;
    WRITELN('CAR.      FREQUENCE       CODE');
    WRITELN('----      ---------       ----')
  END;
END;



{$A-}
PROCEDURE PRINTTREE(VAR ROOT:NODETYPE;C:CODE);
{ Impression du codage }
VAR PT1,PT2 :INTEGER;
BEGIN
  WITH ROOT DO
    BEGIN
      IF LEFTCHILD=NIL THEN
      BEGIN
        PRTCODE(LEFTCHAR,OBJET1[LEFTCHAR].FREQUENCE,C+'0');
        IF LEFTCHAR=256 THEN
        BEGIN
          PUTCHAR(TAMPON^,229);
          PUTCHAR(TAMPON^,$FE)    { END OF FILE }
        END ELSE
        BEGIN
          PUTCHAR(TAMPON^,LEFTCHAR);
          PUTCHAR(TAMPON^,$FF)
        END;
        PT1:=PTFICH1;
        PTFICH1:=PT1+2
      END ELSE
      BEGIN
        PUTCHAR(TAMPON^,SUCC((PTFICH1-TABLE) SHR 2));
        PUTCHAR(TAMPON^,0);
        PT1:=PTFICH1;
        PTFICH1:=PTFICH1+2;
        PRINTTREE(LEFTCHILD^,C+'0')
      END;
      IF RIGHTCHILD=NIL THEN
      BEGIN
        PRTCODE(RIGHTCHAR,OBJET1[RIGHTCHAR].FREQUENCE,C+'1');
        PT2:=PTFICH1;
        PTFICH1:=PT1;
        IF RIGHTCHAR=256 THEN
        BEGIN
          PUTCHAR(TAMPON^,229);
          PUTCHAR(TAMPON^,$FE)    { END OF FILE }
        END ELSE
        BEGIN
          PUTCHAR(TAMPON^,RIGHTCHAR);
          PUTCHAR(TAMPON^,$FF)
        END;
        PTFICH1:=PT2
      END ELSE
      BEGIN
        PT2:=PTFICH1;
        PTFICH1:=PT1;
        PUTCHAR(TAMPON^,(PT2-TABLE) SHR 2);
        PUTCHAR(TAMPON^,0);
```

```
        PTFICH1:=PT2;
        PRINTTREE(RIGHTCHILD^,C+'1')
      END
    END
  END;


(*$A+*)
BEGIN
  WRITELN;
  IF PARAMCOUNT=0 THEN
  BEGIN
    WRITELN('Syntax:  NomFichier[/][ [DestDriv]]');
    WRITELN('          Le slash fait afficher les codes binaires');
    WRITELN;
    UNTILFLAG:=FALSE
  END ELSE
  BEGIN
    UNTILFLAG:=TRUE;
    CMDLIN:='';
    FOR I:=1 TO PARAMCOUNT DO CMDLIN:=CMDLIN+PARAMSTR(I)+'_';
  END;
  REPEAT
    LINECOUNT:=0;
    LEN:=0;
    LEN0:=0;
    IF NOT UNTILFLAG THEN
    REPEAT
      WRITE('*');
      READLN(CMDLIN)
    UNTIL CMDLIN<>'';
    FOR I:=1 TO LENGTH(CMDLIN) DO CMDLIN[I]:=UPCASE(CMDLIN[I]);
    I:=POS(' ',CMDLIN);
    IF I=0 THEN
    BEGIN
      NOM0:=CMDLIN;
      CMDLIN:=''
    END ELSE
    BEGIN
      NOM0:=COPY(CMDLIN,1,I-1);
      WHILE (CMDLIN[I]=' ') AND (I<LENGTH(CMDLIN)) DO I:=I+1;
      CMDLIN:=COPY(CMDLIN,I,2)
    END;
    I:=LENGTH(NOM0);
    IF NOM0[I]='/' THEN
    BEGIN
      NOM0:=COPY(NOM0,1,I-1);
      PRTFLAG:=TRUE
    END ELSE PRTFLAG:=FALSE;
    OPENIN;
    IF IORESULT<>0 THEN
    BEGIN
      WRITELN('File not found');
      CLOSE(FICH0)
    END ELSE
    BEGIN
      LEN0:=FILESIZE(FICH0)*128.0;
      GETFREQ(N);
      CRC0:=CRC;         (* Sauver le ChekSum *)
      CLOSE(FICH0);
      OPENIN;
      NOM1:='';
      CH:=POS(':',NOM0);
      FOR I:=CH+1 TO LENGTH(NOM0) DO NOM1:=NOM1+NOM0[I];
      NOM0:=NOM1;
      CH:=POS('.',NOM1);
      IF CH=0 THEN NOM1:=NOM1+'.Q'
      ELSE IF CH>=LENGTH(NOM1)-1 THEN NOM1:=NOM1+'Q'
      ELSE NOM1[CH+2]:='Q';
      NOM1:=CMDLIN+NOM1;
      WRITELN(NOM0,'===>',NOM1);
      OPENOUT;
      TRI(N);   (* TRIER LES OBJETS *)
      NEW(PROOT);
      TREE(PROOT^,N);  (* CONSTRUIRE L'ARBRE DE CODAGE *)
      FOR I:=0 TO NM0 DO CODAGE[I,0]:=0;
      PUTCHAR(TAMPON^,$76);     (Ce mot est l'identificateur des fich. compr. )
      PUTCHAR(TAMPON^,$FF);
      PUTCHAR(TAMPON^,LO(CRC0));       (Mot reserve pour le CRC)
      PUTCHAR(TAMPON^,HI(CRC0));
      FOR I:=1 TO LENGTH(NOM0) DO PUTCHAR(TAMPON^,INTEGER(NOM0[I]));
      PUTCHAR(TAMPON^,0);       (Nom original du fichier. (0==>fin))
      TABLE:=PTFICH1+2;
      PTFICH1:=TABLE;
      IF PRTFLAG THEN
      BEGIN
        CLRSCR;
        WRITELN('CAR.     FREQUENCE      CODE');
        WRITELN('----     ---------      ----')
      END;
```

```
    PRINTTREE(PROOT^,'');      (* IMPRIMER LE CODAGE *)
    RELEASE(PROOT);
    IF PRTFLAG THEN WRITELN;
    LEN:=LEN+4+LENGTH(NOM0)+1+2+4*(N-1);
    IF LEN>=LEN0 THEN
    BEGIN
      WRITE('Il n''y aura pas de reduction de taille. Continuer ? (O/N)');
      READLN(CARACT);
      IF UPCASE(CARACT)<>'O' THEN
      BEGIN
        CLOSE(FICH1);
        ERASE(FICH1);
        GOTO PROMPT
      END
    END ELSE WRITELN((1-LEN/LEN0)*100:5:2,'% de reduction');
    TEMP:=MEMAVAIL;
    IF TEMP<0 THEN TEMP:=TEMP+65536.0;
    TEMP:=TEMP+1152;
    IF TEMP>=MAXINT THEN TEMP:=MAXINT-1;
    TAILTAMPON:=ROUND(TEMP);
    LENTAMPON:=TAILTAMPON DIV 128;
    TAILTAMPON:=LENTAMPON*128;
    GETMEM(PROOT,TAILTAMPON-1152);      { Restorer les places  }
    I:=PTFICH1;
    PTFICH1:=TABLE-2;
    TABLE:=(I-TABLE) DIV 4;
    PUTCHAR(TAMPON^,TABLE MOD 256);     {Ce mot=nombre d'entree de la table}
    PUTCHAR(TAMPON^,TABLE DIV 256);
    PTFICH1:=I;
    { Transformation des donnees }
    SAVEBITS:=0;
    PTBITS:=1;
    WHILE GETCHAR(CH) DO PUTBITS(CODAGE[CH,1],CODAGE[CH,0]);
    PUTBITS(CODAGE[256,1],CODAGE[256,0]);
    CLOSEOUT(TAMPON^);
    PROMPT: CLOSE(FICH0)
  END
 UNTIL UNTILFLAG
END.
```

Dans cette nouvelle rubrique, qui nous l'espérons sera régulière et fournie, vous trouverez toutes les questions que vous vous posez concernant la programmation, les langages, les systèmes, les réseaux, la technologie.

Il tient à vous, adhérents de JEDI, d'y apporter une réponse. Toutes les réponses reçues seront publiées dans cette rubrique.

QUESTION 1 de Mr RIBOULET (33610 CESTAS)
Comment effectue-t-on le calcul du module en nombre complexe sous FORTH (racine de xE2+yE2). Le but de cette question est de permettre la création d'un programme de calculs en nombres complexes.

QUESTION 2 de J.Y. MALEGEANT (44000 NANTES)
Concernant l'HECTOR HRX:

- comment programmer les équivalences INP et OUT du BASIC 3X pour les ports du I 8255 (problèmes d'adresses?).

- comment dans un programme FORTH obtenir la bascule majuscules/minuscules.

Illustration de couverture et ci-contre réalisées sur THOMSON TO7-70 à l'aide de COLORPAINT

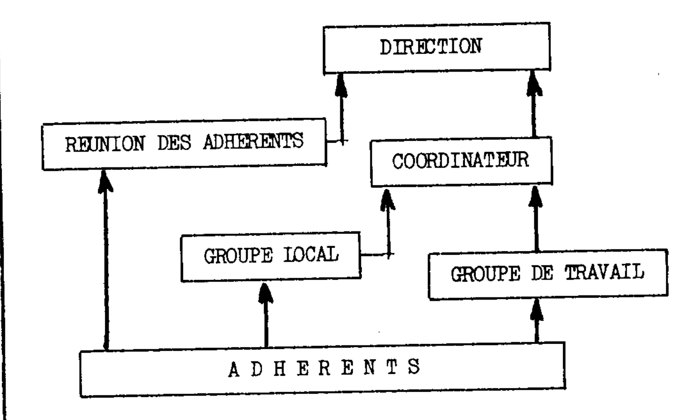


FORTH

Nos contacts avec le groupe des utilisateurs du langage FORTH en RFA deviennent de plus en plus suivis. Après divers échanges de courrier et de coups de téléphone, nous sommes en mesure de vous apporter des précisions sur une association très active et avec laquelle nous avons convenus d'échanger le plus régulièrement possible des informations, des produits et des articles.

UN COUP D'OEIL SUR LE GROUPEMENT FORTH.

Structure:
Le groupement des utilisateurs de FORTH a son siège social à HAMBOURG (HAMBURG). De là sont coordonnées les diverses activités, l'expédition du magazine 'VIERTE DIMENSION' et résolues les questions. Ici s'y trouve aussi la direction qui organise la réunion annuelle des adhérents. Le "vrai" travail concernant FORTH est réalisé par les groupes locaux et les groupes de travail. Chacun dispose d'un coordinateur, lequel rend compte des activités.



Historique:
Début 1984, le groupement des utilisateurs du langage FORTH (BGB) s'est constitué dans le nord de l'Allemagne à partir de 24 Forthiens. Fin 84, une association s'est constituée, dont la réputation s'est consolidée fin 85. Fin 86, le groupement des utilisateurs du langage Forth compte environ 200 membres à travers toute l'Europe.

Buts:
Le but essentiel du groupement Forth est de faire connaitre le langage Forth ainsi que les principes de programmation qui en découlent. Ceci suppose des cours de formation, des réunions d'informations, ainsi que des travaux de groupe sur des projets d'intérêt général.

Activités:
Le groupement des utilisateurs de Forth coordonne les activités, diffuse les informations provenant d'un utilisateur ou d'un groupe de travail. Ci-joint un coup d'oeil sur les principales activités. Les détails sont fournis par les personnes dont les adresses figurent ci-après. Les initiatives pour une extension des activités sont les bienvenues!

VIERTE DIMENSION (QUATRIEME DIMENSION) Magazine Forth
La seule publication Forth en langue allemande diffusant les documents, articles et références pour contact.
Rédaction: Michael KALUS (notre contact no1 pour JEDI)
Praesidentenstr.40, D-5830 SCHWELM, tel: (19-49) 02336-82204 (ne pas composer le '0' à partir de la France)

volksFORTH83 groupe de travail
Forth pour C64, ATARI ST et SCHNEIDER (AMSTRAD) CP/M (MSDOS en cours... mais JEDI est sympa et leur en a envoyé un exemplaire).
Contact: Bernd PENNEMANN (notre contact no2 pour JEDI)
Steilshooper Str.46, D-2000 HAMBURG 60, tel (19-49) 040-6900539

ForthTREE (l'arbre FORTH)
Le système d'information et de communication du Groupe Forth (boite aux lettres). Pour contact téléphonique: (19-49) 040-3904204 (300 + 1200 bd). Autres systèmes locaux en cours d'étude.

SysOp: Marco PAUCK
Friensallee 92, D-2000 HAMBURG 50, Tel: (19-49) 040-3900139

La ligne service du bureau Forth
Pour toute information, question et renseignements. En ligne directe avec nos experts. Tous les mardis de 18-20h au (19-49) 040-3904204

Bibliothèque
Au siège social sur tout ce qui concerne Forth et consultable par tous les membres.

Service copie d'articles et programmes pour tous les membres.
Contact: Thomas PRINZ
Adalbert-Stifer-Str.2, D-6930 EBERBACH a/N, tel: (19-49) 06271-2830

euroFORML
Réunion annuelle des experts Forth du groupe européen 'Forth Modification Laboratory (FORML)' en provenance d'Europe et des USA.
Contact: Klaus SCHLEISIEK
Steinberg 8a, D-2000 WEDEL, tel: (19-49) 04103-13255.

L'ARBRE FORTH (ForthTREE)

C'est le système de communiaction du FIG HAMBURG. Pour les adhérents de JEDI nous avons fait un essai de communication afin d'évaluer l'intérêt des services et informations proposées. Nous y avons trouvé un serveur (300-1200bds) d'une qualité et d'une facilité de consultation qui mérite d'être signalé. Aucun service n'est restreint. Bien entendu, il est impératif d'avoir des notions de langue allemande. Nous avons enregistré une communication sur ForthTREE dont vous trouverez la transcription ci-après. Chaque ligne est le reflet des informations telles que nous les avons reçues. Pour plus de commodité, nous avons traduit l'essentiel. Cependant, nos explications complémentaires seront indiquées entre crochets [comme ceci].

[composer le numéro 19 ..tonalité grave.. 49
puis 40 39004204 .. sonnerie .. porteuse .. et connexion]

Appuyer sur <RETURN> svp

Bienvenue dans la base de données forth
du groupement des utilisateurs du langage FORTH

La base de données a commencé ses activités le 28-AVR-86.

Les nouveaux arrivants utiliseront la commande

READ EINFUEHRUNG

pour avoir les informations de base leur permettant de travailler sur la base de données.

Attention: la base de données est inactive le mardi de 18-20h. Durant ce laps de temps, vous serez en ligne directe avec un de nos animateurs pour toute information.

[nous vous donnons ici les informations permettant de manipuler ForthTREE afin de diminuer le temps de communication]

READ HILFE pour obtenir de l'aide à tout moment
READ KONFERENZEN pour commencer

<CNTRL-S> stop, <CNTRL-Q> continuer
<C> interrompre, <N> passer et continuer

*** 12-NOV-86 HILFE
nouvel arrivant: KONFERENZEN
messages lus: 411

<CNTRL-S> stop, <CNTRL-Q> continuer

Seules les commandes et options, tel B)EFEHL listées ci-après, au format abrégé sont valides. Les informations <noms> ne peuvent être abrégées.

Les commandes principales:

R)EAD <name> — affiche une information <nom>.
B)ROWSE <name> — affiche la première partie d'une information <nom>.
I)NDEX <name> — affiche une liste de toutes les informations suivantes.
P)ARENT — montre au débutant l'information en référence arrière.
A)DDTO <name> — rajoute une information à celle nommée <nom>.

LOGOFF, BYE pour interrompre la liaison.

<C> ou <X> interrompent un R)EAD, B)ROWSE ou I)NDEX.
<K> ou <N> interrompent un R)EAD ou B)ROWSE et laissent l'accès à l'information suivante, sous réserve de modification des options C)OMPLETE ou S)TARTING.

I HILFE — liste les menus d'aide détaillés.
R HILFE C — montre tous les menus d'aide afférents à l'information courante.

R EINFUEHRUNG — guide d'utilisation pédagogique.

R KONFERENZEN — revient au sujet de départ.

Informations suivantes:

```
.04-AUG-86  BEFEHLE      COMMANDES
.04-AUG-86  OPTIONEN     OPTIONS
```

[Ici nous avons essayé une commande au hasard. Quelle chance, ForthTREE avait la référence dans son catalogue]

Befehl? I FORTH

[La partie française est de notre cru. Elle permet aux non germanistes d'avoir une idée du contenu de la base de données afférente à FORTH dans ForthTREE. Cette liste peut paraitre un peu longue, mais elle permettra à ceux qui consulteront cette base d'écourter le temps de connexion, et leur facture téléphonique, en troouvant immédiatement la référence qui les intéresse. Exemple:

Befehl? R NOVIX4000

vous liste le contenu de l'article faisant référence au microprocesseur NOVIX 4000.]

```
02-JUN-86  FORTH                          FORTH
.14-DEC-86  FORTH.GESELLSCHAFT             Groupement FORTH
..31-DEC-86  LOKALE.GRUPPEN                Groupes locaux
...28-JUL-86  HAMBURG                      HAMBOURG
....11-JUN-86  TERMINE86
....24-JAN-87  TERMINE87
...22-MAY-86  KARLSRUHE                    KARLSRUHE
...19-OCT-86  WUPPERTAL                    WUPPERTAL
....19-OCT-86  THEMEN86                    Thèmes 86
...30-JUL-86  DARMSTADT                    DARMSTADT
...20-NOV-86  MUENCHEN                     MUNICH
...31-DEC-86  PADERBORN                    PADERBORN
...22-MAY-86  BERLIN                       BERLIN
...22-MAY-86  HANNOVER                     HANNOVRE
...30-JUL-86  FREIBURG                     FRIBOURG
...22-MAY-86  FRANKFURT                    FRANCFORT
...31-DEC-86  STUTTGART                    STUTTGART
...25-JAN-87  MAINZ                        MAYENCE
..31-DEC-86  FACHGRUPPEN                   GROUPES DE TRAVAIL
...22-MAY-86  VOLKS/ULTRA-FORTH            idem
...31-DEC-86  GRAPHIK                      GRAPHISME
...31-DEC-86  32-BIT-FORTH                 FORTH 32 BITS
...31-DEC-86  FORTH-MASCHINEN              MACHINES FORTH
...25-JAN-87  KI                           KI
...25-JAN-87  DATENKOMMUNIKATION           TRANSMISSION DONNEES
..31-DEC-86  FG-INTERNATIONAL              Groupes Forth INTERN.
...31-DEC-86  HOLLAND                      HOLLANDE
...31-DEC-86  OESTERREICH                  AUTRICHE
...25-JAN-87  SCHWEIZ                      SUISSE
...25-JAN-87  ENGLAND                      GRANDE BRETAGNE
...25-JAN-87  BELGIEN                      BELGIQUE
...25-JAN-87  IRLAND                       IRLANDE
...25-JAN-87  ITALIEN                      ITALIE
...25-JAN-87  FRANKREICH                   FRANCE
....31-DEC-86  FRONKROISCH                 ????
..19-OCT-86  FORTH.MAGAZIN                 MAGAZINE FORTH
...17-JUN-86  GAST.EDITORIAL               EDITORIAL INVITE
...21-SEP-86  INFO-HEFT3                   CAHIER INFOS 3
...07-JAN-87  INFO-VII/NO.4                INFO VII/No 4
..14-DEC-86  MITGLIEDSCHAFT                TRAVAUX DES ADHERENTS
...14-OCT-86  JAHRESTREFFEN.86             REUNION ANNUELLE 86
....14-OCT-86  TAGESORDNUNG                ORDRE DU JOUR
....09-DEC-86  PROTOKOLL.MV86              PROTOCOLE MV 86
...11-NOV-86  MIKROMODUL                   MICRO MODULE
....11-NOV-86  JAHRESVERSAMMLUNG           RASSEMBLEMENT ANN.
..11-JUN-86  BUEROZEITEN                   HORAIRES DU BUREAU
.07-MAY-86  FORTH.SYSTEME                  SYSTEME FORTH
..22-APR-86  FIG                           FIG
...27-OCT-86  FIG-BUGS                     BUGS du FIG
..18-APR-86  F83                           F83
...23-MAY-86  F83.WO                       F83.WO
....11-JUL-86  F83.APPLE                   F83.APPLE
.....11-JUL-86  ANT:F83.APPLE              REPONSE:F83.APPLE
...28-AUG-86  F83-SOFTWARE                 LOGICIELS F83
....23-NOV-86  FULL-SCREEN-EDITOR          EDITEUR PLEIN ECRAN
...28-DEC-86  V24.AN.CP/M.F83              TRANSM V24 F83/ CP/M
...06-JAN-87  F83-IBM-SOURCE               SOURCE F83 POUR IBM
....06-JAN-87  META86.BLK                  META86.BLK
....06-JAN-87  KERNEL86.BLK                KERNEL86.BLK
....06-JAN-87  EXTEND86.BLK                EXTEND86.BLK
....06-JAN-87  CPU8086.BLK                 CPU8086.BLK
....06-JAN-87  UTILITY.BLK                 UTILITY.BLK
....06-JAN-87  HUFFMAN.BLK                 HUFFMAN.BLK
....06-JAN-87  CLOCK.BLK                   CLOCK.BLK
....06-JAN-87  F83-FIXS.TXT                MISES A JOUR F83
...15-JAN-87  F83-IBM-SOURCE???            SOURCE??? F83 IBM
....16-JAN-87  METACOMPILATION...          METACOMPILATION...
.....16-JAN-87  WAR NICHT?                 POURQUOI PAS?
......17-JAN-87  OBJECTCODE-TRANSFER       TRANSFERT CODE OB
.......18-JAN-87  FILECONV.EXE?            FILECONV.EXE
........18-JAN-87  TESTFILE                TESTFILE
..21-APR-86  VOLKSFORTH                    VOLKSFORTH
...21-APR-86  VOLKS.NEWS                   NOUVELL. VOLKSFORTH
....10-MAY-86  VOLKSFORTH.UPDATE.1         M.A.J FOLKSFORTH
...11-MAY-86  DICTIONARY.FULL?             DICTIONNAIRE PLEIN?
...28-MAY-86  CASSETTE?                    CASSETTE?
....28-MAY-86  ANTW.CASSETTE               REPONSE A CASSETTE
...08-SEP-86  VHS-KURS                     COURS SUR VHS
....12-NOV-86  VDI-SEMINAR                 SEMINAIRE VDI
...26-OCT-86  PUBLIC.DOMAIN?               DOMAINE PUBLIC?
...26-OCT-86  AERGERLICH?                  CRISPE?
.....26-OCT-86  UMSTAENDLICH!              COMPTE RENDU!
...10-NOV-86  VERSION-3.80!!               VERSION 3.80!!
....27-DEC-86  NEUES.HANDBUCH              NOUVEAU MANUEL
..28-JUN-86  TI-99/4A?                     TI-99/4A?
...30-JUN-86  TI-99/4A!!!                  TI-99/4A!!!
..11-JUL-86  WO.APPLE-FORTH                WO pour FORTH APPLE
...15-JUL-86  APPLE-FORTH.HIER             ICI pour FORTH APPLE
..14-OCT-86  SIRIUS-PC?                    PC?-SIRIUS
.01-JUN-86  FORTH.MASCHINEN                MACHINES FORTH
..22-APR-86  WINFIELD                      WINFIELD
..22-APR-86  BENCHMARKS                    TESTS DE PERFORMANCES
..26-MAY-86  SYMBOLIC.CONTROL              CONTROLE SYMBOLIQUE
...15-SEP-86  MEHR-INFOS?                  PLUS D'INFORMATION?
....16-SEP-86  ALLE-INFOS!                 TOUTES LES INFOS
..30-JUL-86  NOVIX-CHIP                    PUCE NOVIX
...30-JUL-86  NOVIX-LETTER                 LA LETTRE NOVIX
...01-AUG-86  DELTA-BOARD                  CARTE DELTA
....01-AUG-86  DELTA-BOARD-FRAGEN          QUESTIONS CAR.DELTA
.....02-AUG-86  DELTA-BOARD-ANTWORT        REP./ CARTE DELTA
....16-SEP-86  FORTHKIT                    KIT FORTH
...15-SEP-86  NOVIX.IN.HAMBURG             NOVIX A HAMBOURG
...28-SEP-86  V4000                        V4000
...04-OCT-86  NEUE-CHIPS                   NOUVELLES PUCES
...04-NOV-86  NOVIX-EB1                    NOVIX EB1
...04-NOV-86  NOVIX-CLUB                   CLUB NOVIX
.07-MAY-86  FORTH.FRAGEN                   QUESTIONS SUR FORTH
..07-MAY-86  FORTH-ANLEITUNG?              CONSEIL FORTH?
..16-MAY-86  FORTH?                        FORTH?
..04-SEP-86  8085-FORTH-HEX-LISTN          LISTING HEX FORTH8085
..05-SEP-86  HEX-LISTING?                  LISTING HEXA?
..06-OCT-86  F83-LEHRBUCH                  OUVRAGE DEBUTANT F83
...07-OCT-86  ANT:F83-LEHRBUCH             REP: OUVR.DEB. F83
....18-OCT-86  BEZUGSQUELLE                TRAITEMENT SOURCES
..22-JAN-87  FORTH BUECHER                 LIVRES FORTH
...25-JAN-87  ANT:FORTH BUECHER            REP: LIVRES FORTH
..26-JAN-87  FORTH.BUECHER                 LIVRES FORTH
.31-AUG-86  FORTH.CODE                     CODE FORTH
..02-JUN-86  WORT.DES.MONATS£1             ???
...26-NOV-86  WORT.DES.MONATS£2            ???
..31-AUG-86  FIG-LIBRARY                   LIBRAIRIE
..31-AUG-86  EXPLAIN                       EXPLICATIONS
..04-SEP-86  RANDOM-NUMBERS                NOMBRES ALEATOIRES
..04-SEP-86  TRANSIENT-DICTIONARY          DICT. TRANSITOIRE
..06-SEP-86  MINI-TREE                     MINI ARBRE
....06-SEP-86  FBBS                        FBBS
....06-SEP-86  FBBS2                       FBBS2
..31-AUG-86  DOWNLOAD-PROTOKOLL            PROTOCOLE DOWNLOAD
..01-SEP-86  TERMINALPROGRAMM              PROGRAMME DE TERMINAL
```

..03-SEP-86 GEM.MIT.VOLKSFORTH GEM AVEC FORTH
..17-SEP-86 CHUCK-MOORE-FORTH FORTH CHUCK MOORE
...06-JAN-87 CHUCK-MOORE-FORTH2 FORTH2 CHUCK MOORE
..17-SEP-86 APPLICATION+RESEARCH APPLICATIONS+RECHERCHE
..16-JAN-87 QUICK-TEXT-FORMATTER FORMATAGE TEXTE RAPIDE

[Ici nous avons fait un essai de lecture d'un sujet]

Befehl? R F83-FIXS.TXT
*** 06-JAN-87 F83-FIXS.TXT
Vorgaenger: F83-IBM-SOURCE
gelesen: 21

LIST F83-FIXS

This file describes most of the changes to F83 between versions
1.0 and 2.0.

-It is always difficult to follow a moving target. In the six months since we released version 1.0 we have received so many good suggestions that the temptation to use some of them was impossible to resist. To all of you who contributed, thank you again. We will try to avoid any further changes until 1985 at the earliest. If there are bugs, we will report them separately. Updating the various versions is a lot of work even without offering any support, and we are tired. It is time to move on to applications, and do something useful for a change.

-The changes were as follows:

General:
* Removed the superfluous NOOP from all self-defining words.
* Changed all instances of C; to END-CODE (by request).
* Partitioned META into META.BLK (the meta-compiler) and KERNEL.BLK (the source for the kernel).

META:
* Fixed .SYMBOLS

KERNEL:

* Removed null from the system. Sealed search orders no longer require the old magic null word.
* Fixed PARSE and PARSE-WORD. They used to increment >IN past the end of source text.
* Changed CP/M to DOS.
* Moved kernel DOS words into DOS vocabulary.
* Added USER VARIABLE IN-FILE. All file operations read from IN-FILE and write to FILE. This allowed removing the confusing FILES vocabulary. User interface is unchanged: FROM <file> makes <file> the IN-FILE. OPEN <file> makes both the same. LOAD uses IN-FILE, then resets it to FILE. This is probably appropriate.
* FBLOCK and FBUFFER take an fcb address and a block number.
* SWITCH exchanges FILE and IN-FILE.
* ?UPPERCASE conditionally forces a string to upper case. Used by DEFINED and FORGET.
* EMIT primitives renamed: (CONSOLE) is console only, (EMIT) is for console and maybe also printer, depending on PRINTING.
* Fixed CONTROL.
* Made default (PRINT) not use LISTST, because it hangs on many systems. Optionally use LISTST if available for faster spooling.
* Renamed FORTH control character table from CC1 to CC-FORTH.
* Changed DO to ?DO in -TRAILING.
* Deleted HEADER from CREATE, made CREATE do it all.
* Changed ,VIEW to make file 0 if BLK is 0.
* Added ςS for comment to end of screen.
* Added better error handling for disk reads and writes.
* Accessing a BLOCK which is Out of Range no longer leaves the buffer assigned to the non-existent block.
* Changed DISCARD to mark discarded buffer as empty.
* .FILE and FILE? added to display file names.

EXTEND:
* Split ONLY into the ONLY operator and the ROOT vocabulary.
* Removed OPEN-FILE from FILE: and added it to VIEW.
* Added VIEWS which installs files into VIEW-FILES table.
* Moved SET-DRIVE into EXTEND, changed it to use the BIOS to determine whether a drive is legal, and renamed it SELECT.
* DRIVE? prints the current drive.

Und nun? R F83-SOFTWARE

*** 28-AUG-86 F83-SOFTWARE
Vorgaenger: F83
gelesen: 49
Einen ganzen Haufen Forth Programme fuer das F83 findet man in einer Library, die von John A. Peters betreut wird. Der dort gesammelte Forth-Code ist allerdings von unterschiedlicher Qualitaet. Einige der Programme und Tools aus der Library werden wir im Zweig FIG-LIBRARY veroeffentlichen.
[Une grande quantité d'ouvrages FORTH sont disponibles à la librairie tenue par John A. Peters. Les divers codes Forth sont disponibles. Divers programmes et outils de cette librairie sont diffusés dans la branche FIG-LIBRARY.]

LIBRARY MANAGED BY:

John A. Peters
121 Santa Rosa Ave.
San Francisco, CA 94112

Phone (415) 239-5393
8-9 am or after 7:30 pm
or week ends.

These screens are 83 standard FORTH Laxen/Perry Model 2.1.1. They may be copied and improved, but they may not be sold. SHARE WARE IS A TWO WAY STREET. Upload a file you improved etc. If you found a file(s) you like, donate $7.00 to FIG Library. Write to me c/o FIG Tree BBS (415) 538-3580 300 bps (Text) or CL BBS (415) 957-9370 300/1200 (Files and E-Mail) New ideas?
[Ici le numéro pour se connecter à la librairie FORTH en 300/1200 bauds aux USA!!!. Les écrans sources ainsi diffusés sont libres de tout droit mais ne peuvent être vendus. Si vous trouvez un programme qui vous plait, envoyez la somme de 7.00$ à la Librairie FIG -principe du SHARE WARE-]

Folgenachrichten: [Information(s) suivante(s)]
.23-NOV-86 FULL-SCREEN-EDITOR

Befehl? R FRANKREICH

*** 25-JAN-87 FRANKREICH
Vorgaenger: FG-INTERNATIONAL
gelesen: 8 [signifie le nombre de fois où une info a été consultée]

Franzoesisches FIG-Chapter

Kontakt:

Jean-Daniel Dodin
77 Rue du Cagire
F-31100 Toulouse
Tel.: (16-61)44-03

Folgenachrichten:
.31-DEC-86 FRONKROISCH [Là on n'a pas compris et on ne cherche pas à comprendre. Ce sera pour une autre fois.]

Befehl? BYE [Ici on a décidé de décrocher, parce que en 300 bds, bonjour la facture...]

Vielen Dank fuer Ihren Anruf!
[Merci beaucoup pour votre appel!]

COMMENTAIRE

Enfin un serveur FORTH digne d'intérêt en Europe. A noter que la connexion avec la RFA (1 taxe/10 sec) n'est guère plus coûteuse qu'une communication interurbaine (1 taxe/12 sec). Certaines informations sont du plus grand intérêt, nottament celles concernant le NOVIX4000 et qui sont en anglais. En outre, on y trouve les coordonnées de groupes de travail et de personnes compétentes. Pour ceux qui ne pratiquent pas l'allemand, ils pourront communiquer avec la majorité des animateurs du FIG HAMBOURG en langue anglaise.

RACINE CARREE 16/32 BITS

par FIG HAMBOURG

Dans le numéro 27, nous diffusions une première version du calcul de l'extraction de la racine carrée d'un nombre. Le magazine VIERTE DIMENSION posait ce problème à titre d'exercice aux membres de FIG-HAMBOURG. Voici le compte rendu des réponses obtenues.

Résolution de l'exercice du précédent magazine: le mot s'appelait WURZEL. Nous ont répondu:

(1) Finn Berlev, Lillevangsvey 92, DK-3520 FARUM, DANEMARK

(2) Ulrich Hoffman, Harmsstrasse 71, D-2300 KIEL 1, BRD

(3) Joh. Polster, Speerstrasse 7, CH-8820 Wädenswil, Suisse.

Bonne chance au nom du groupement des Utilisateurs de Forth. La formule à appliquer était: (n+1)^2=n^2+(2n+1).

Mr Polster commentait la pile et l'algorithme par:
WURZEL (n-au-carre — chaîne)

Mr Hoffman avait une autre proposition pour le nom de cette fonction. Il écrit:" le plus évident est SQR, mais en Pascal on préfère SQRT. Personnellement, je penche pour CRT (pour: carrot) ou en bon allemand MHRB (pour: Mohrrube - en français carotte) et me déterminais finalement pour l'icone v- comme nom.

Mr Berlev résume le problème par SQRT. "Ce problème me rappelle le temps, il y a déjà longtemps, où j'apprenais à faire ce calcul à la main..." et commente DSQRT par "...n'est pas le plus élégant, mais peut être utilisé pour des nombres double précision et délivre le reste."

```
: SQRT ( limit — n )  1 swap 0 ?do 2+ dup +loop 2/ ;

: (DSQRT1)  ( ud — a0 a1..an - 1 n )
1 begin >r 4 ud/mod 2dup or
  while r> 1+
  repeat 2drop 1- r> ;

: (DSQRT2)  ( a0 a1..an - 1 n — ur uq)
1 under ?do
  2* -rot 4 * + over 2*
    2dup u> if    - 1- swap 1+
            else drop swap      then loop ;

: DSQRT  ( ud — ur uq )
  2dup d0= ?exit (dsqrt1) (dsqrt2) ;
```

Glossaire

SQRT n1 — n2
"squareroot"
Prend un nombre n1 sur la pile et calcule la racine carrée de ce nombre en tant qu'entier 16 bits sans le reste.

DSQRT ud — ur uq
"d-squareroot"
Prend un nombre double précision non signé sur la pile et calcule la racine carrée uq et le reste ur tels que ud=uq^2+ur.

NOVIX 4000

FIG WUPPERTAL

Autre information nous parvenant d'Allemagne. Elle concerne le nouveau microprocesseur NOVIX 4000 dont voici les caractéristiques (extrait du cahier du groupement d'utilisateurs Forth de WUPPERTAL, animateur Michael KALUS).

- processeur monochip HCMOS programmable en Forth.
- interprétation directe de la majorité des primitives Forth en un cycle machine.
- temps de cycle: 125 nano secondes.
- admet un adressage 64K. Passage à 4M en adressage étendu.
- instructions IF, ELSE, et LOOP exécutables en un seul cycle.
- extraction de racine, multiplication et division en un seul cycle.
- accès à la mémoire locale (fetch et store) en un seul cycle.
- deux ports bi-directionnels.

Le uP NC4000P est le premier membre de la famille NOVIX d'une gamme de microprocesseurs à grande vitesse. L'intérêt du NC4000P est d'intégrer des instructions de haut niveau de manière identique à des instructions machines.

Le NC4000P semble particulièrement adapté au traitement de données en temps réel. Pour indication, voici un aperçu des performances du NC4000P.

Boucle à vide 0 à 1 000 000

68000 (8MHZ) polyForth	18.0 s
68000 (8MHZ) Assembleur	7.0 s
Intel 80286 (10MHZ) C	5.0 s
NC4000P (6MHZ) novixFORTH DO..LOOP	2.4 s
MF16LP (20MHZ) FORTH	1.0 s
NC4000P (6MHZ) novixFORTH FOR..NEXT	0.17 s

Crible d'Eratosthène x 10

68000 (8MHZ) polyForth	29.0 s
68000 (8MHZ) Assembleur	4.9 s
Intel 80286 (10MHZ) C	6.6 s
VAX 780 C	1.4 s
CRAY-1 FORTRAN	1.1 s
MF16LP (20MHZ) FORTH	1.09 s
NC4000P (6MHZ) novixFORTH (cell)	0.85 s
NC4000P (6MHZ) novixFORTH (optimisé)	0.45 s
IBM 3033 PL/I	0.36 s
IBM 3081 PL/I	0.34 s

Suite de Fibonacci à 24

8088 (IBM PC) PolyFORTH	19.0 s
68000 (8MHZ) PolyFORTH	9.5 s
SUN 2/120 (68K) C	1.7 s
INTEL 80286/310 (10MHZ) C	1.2 s
NC4000P (6MHZ) novixFORTH	0.19 s

L'ensemble de développement NC4000P se nomme FORTHkit's.

Contact: COMPUTER COWBOYS, 410 STAR HILL ROAD, WOODSIDE, CA-94062 (USA) tel: (19-1) 415-851 4362

Attention, l'équipe de MOORE qui a déjà réalisé le NOVIX 4000 planche sur un nouveau produit qui devrait voir le jour début 87, le NC6000, dont voici en avant première quelques caractéristiques:

- machine 16 bits
- 2 uCMOS
- 9 MHZ
- 14 Mips

à suivre...

VIERTE DIMENSION



par Gérard SOULA

OPERATEURS D'ENTREE ALPHA-NUMERIQUE

REFERENCES:
Difficulté de programmation moyenne
Catégorie chaînes de caractères
Difficulté d'exercice facile

L'EXERCICE:
Forth permet, à partir des mots de base, de créer toute sorte d'opérateurs de saisie alpha-numériques.

LE PROGRAMME:
D'abord quelques explications sur ces mots qui peuvent être ou non présents selon les versions de Forth.

BLANK (ad n —) parfois nommé BLANKS remplit de caractères 32 n octets à partir de l'adresse ad.

WORD (c — ad) saisie d'une chaîne dans le tampon d'entrée jusqu'à un délimiteur c (code ASCII) ou sur une longueur forfaitaire variable selon les systèmes (40, 72, 80). Puis il transfère cette chaîne à HERE dont il fournit l'adresse en mettant dans le premier octet la longueur. Certaines versions de FORTH ne fournissent pas cette adresse.

TEXT (c —) transfère la chaîne saisie dans PAD avec: soit en mettant la longueur de la chaîne dans le premier octet, soit en mettant directement la chaîne.

Mais ces mots ne permettent de saisir des chaînes qu'à la volée si on veut saisir lors de l'exécution d'un programme. On utilisera par exemple QUERY pour définir PUT. On peut aussi vouloir saisir une chaîne et la transférer à une adresse spécifiée sur une longueur maximale: $PUT

EXEMPLES D'UTILISATION:

SALUT vous demande votre nom et dit bonjour. L'affichage diffère selon la version de TEXT.
PAD COUNT TYPE si la longueur est dans PAD.
PAD 80 - TRAILING TYPE dans le cas contraire.

?IDENTITE saisit le nom et le prénom dans deux variables. Il y a remise à blanc de la variable sur la longueur maximale et contrôle de la longueur à transférer pour ne pas aller écraser le dictionanire.

.IDENTITE affiche le nom et le prénom.

EXTENSIONS POSSIBLES:
Contrôle de la nature de la chaîne saisie et message d'erreur. gestion d'écran et définition de zones de saisie.

LE LISTING:

```
: BLANK  32 FILL ;
: WORD   WORD HERE ;

: TEXT ( c —)   ( la chaine s'implante à partir de PAD   )
PAD 80 BLANK    ( initialisation de 80 octets            )
WORD            ( saisie de chaîne stockée à HERE         )
COUNT           ( adresse de départ et longueur           )
PAD SWAP        ( adresse de destination                  )
CMOVE ;         ( transfert dans PAD                      )

: TEXT ( c —)   ( la chaine s'implante dans PAD+1         )
PAD 80 BLANK WORD ( PAD contient la longueur              )
PAD OVER Ca  1+  ( on transfère la longueur 1+            )
CMOVE ;

: PUT QUERY 1 TEXT ;

: SALUT
CR ." Quel est votre nom? " PUT
CR ." Bonjour "
PAD 80 -TRAILING TYPE ;

: SALUT
CR ." Quel est votre nom? " PUT
CR ." Bonjour "
PAD COUNT TYPE ;

: $PUT  ( ad lg —)
QUERY 1 WORD
COUNT ROT MIN
ROT SWAP CMOVE ;

CREATE NOM 20 ALLOT     CREATE PRENOM 20 ALLOT

: ?IDENTITE
CR ." Nom? " NOM DUP 20 BLANK 20 $PUT
." Prenom? " PRENOM DUP 20 BLANK 20 $PUT ;

: .IDENTITE  CR
NOM 20 -TRAILING TYPE SPACE
PRENOM 20 -TRAILING TYPE CR ;
```

CHAINES DE CARACTERES

REFERENCES:
Difficulté de programmation moyenne
Catégorie utilitaire
Difficulté d'exercice moyenne

L'EXERCICE:
Il s'agit de se constituer quelques utilitaires de gestion de chaînes de caractères.

LE PROGRAMME:

$ (n — xxx) permet de déclarer une variable alpha-numérique de longueur n, nommée xxx et dont ma structure est la suivante:
- 1er octet contient la longueur maximale
- 2ème octet contient la longueur de la chapqine
- les n octets suivants contiennent la chaîne.

Lorsque le nom xxx est tapé, on obtient l'adresse d'implémentation de la chaîne elle-même, c'est à dire l'adresse de son premier caractère affichable.

$MAX (xxx — lgmax) donne la longueur maximale de la chaîne xxx.

$. (xxx —) affiche la chaîne xxx

$" (xxx —) saisie d'une chaîne à la volée et la transfère dans xxx en contrôlant la longueur.

$! (xxx —) pareil mais arrête le programme.

-TEXT (ad1 lg ad2 — f) compare les chaînes de caractères implantées à ad1 et ad2 sur la longueur lg et rend les paramètres suivant:
f=0 si les chaînes sont égales
f=-1 si ad1 < ad2
f=1 si ad1 > ad2

$= (xxx xxx' — f) compare les chaînes xxx et xxx'

.LEFT (xxx n —) affiche les n caractères à gauche de xxx.
.RIGHT (xxx n —) affiche les n caractères à droite de xxx.

POUR UTILISER L'EXERCICE:

```
20 $ NOM
$" PSI.EDITION
$.  affiche  PSI.EDITION OK
```

LE LISTING:

```
: $ ( n — xxx)
CREATE                       ( crée le nom de la variable)
DUP C,                     ( stocke la longueur maximale)
0 C,                     ( longueur de la chaine présente)
DOES> ( xxx — ad)
  2+ ;             ( adresse d'implantation de la chaine)

: $MAX ( ad — lg)  2- Ca  ;

: $LEN  ( ad — lg)  1- Ca  ;

: $. ( ad —) DUP $LEN TYPE SPACE ;

: $"  ( xxx — yyy)
DUP $MAX  HERE OVER BLANK  1 WORD   ROT 1-  ROT CMOVE ;
```

suite page 20

LE ROBOT MULTISOFT

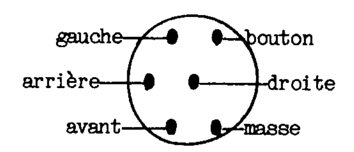
C'est un petit robot entrant dans la catégorie des bras manipulateurs et dont la vocation est essentiellement pédagogique. Il possède six moteurs pas à pas, permettant le découpage en autant d'organes élémentaires:

- la base,
- l'épaule,
- le coude,
- le poignet,
- rotation de la main,
- les doigts.

Il est alimenté en courant continu 12-15V, 5A et la tension des signaux de commande sont aux caratéristiques TTL.

RACCORDEMENT AU THOMSON

Ne disposant pas d'interface parrallèle, le raccordement du robot MULTISOFT s'est effectué sur la prise des manettes de jeu de l'interface jeu/musique. Les signaux issus de cet interface sont compatibles avec les signaux de commande du robot. En effet, l'interface utilise un PIA 6821 que nous pourrons programmer en sortie.



Les points qui nous intéressent sont "arrière, avant, droite, gauche" lesquels correspondent à b0, b1, b2 et b3 pour le connecteur 1, b4, b5, b6 et b7 pour le connecteur 2. Les valeurs b0 à b7 correspondent aux bits b0 à b7 du port A du PIA 6821.

La mise à zéro du PIA port A est assurée par la fonction INITPRA (Pour INITialisation du PoRt A). Le bit de poids faible du port A est mis à 1 pour inhiber les actions du robot. La validation d'une action du robot étant validée par un front descendant sur b0 du PORTA:

```
HEX
E7CC CONSTANT PORTA  E7CC CONSTANT DDRA
E7CE CONSTANT CRA
DECIMAL

: INITPRA ( ——)
   0 CRA  C!
 255 DDRA C!
   4 CRA  C!  ;

INITPRA 1 PORTA C!
```

THOMSON
TO7 - TO7/70

LA COMMANDE DES MOTEURS

Les moteurs sont sélectionnés par le nom de l'organe correspondant. Pour ce faire, six constantes ont été définies: DOIGT, MAIN, POIGNET, COUDE, EPAULE et BASE. Les valeurs affectées à chaque organe correspondent aux valeurs binaires qui seront utilisées pour sélectionner sur le port A la fonction correspondante. Exemple:

```
POIGNET PORTA C!
```

sélectionne le poignet comme étant le prochain organe à commander.

Le listing de la partie activation/désactivation d'un organe est:

```
 2 CONSTANT DOIGT
 4 CONSTANT MAIN
 6 CONSTANT POIGNET
 8 CONSTANT COUDE
10 CONSTANT EPAULE
12 CONSTANT BASE

CREATE PAS
 48 C,  176 C,  144 C,  208 C,
192 C,  224 C,   96 C,  112 C,

VARIABLE ORGANE

: SELCTION  ( n ——)
ORGANE !

: ACTIVATION    ( n1 ——)
254 AND PORTA C!  ;

: DESACTIVATION ( n1 ——)
1   OR  PORTA C!  ;
```

Les différentes positions du moteur sont sélectionnées par les bits b4 à b7 du port A, ce qui explique les valeurs attribuées au tableau PAS. Chacune de ces valeurs doit être complémentée à la valeur de l'organe courant et la valeur de commande d'activation ou de désactivation pour faire tourner le moteur. Ces valeurs doivent être envoyées dans l'ordre où elles sont définies dans PAS:

```
0 PAS + C@ POIGNET + 254 AND PORTA C!
0 PAS + C@ POIGNET +   1  OR PORTA C!
1 PAS + C@ POIGNET + 254 AND PORTA C!   etc...
```

provoque la rotation du moteur du poignet d'un huitième de tour.

Il faut aussi tenir compte du fait qu'un moteur pas à pas possède une certaine inertie mécanique lors des accélérations et décélérations. Une temporisation variable sera appliquée entre la commande de chaque pas du moteur de l'organe en mouvement. Divers essais ont montré qu'une temporisation optimale de 10 ms semblait nécessaire, ce qui permet une vitesse de rotation de (100 pas)/8, soit 750tr/mn. Cette vitesse ne peut être appliquée directement, d'où les fonctions ACCELERATION et DECELERATION ont une montée et une descente en vitesse exponentielle.

Les mots ALLERS et RETOURS activent un nombre de pas n sur l'organe sélectionné:

```
BRAS SELECTION 100 ALLERS
COUDE SELECTION 250 RETOURS  etc...
```

La mise au point des différentes valeurs de temporisation initiale et terminale ont demandé l'application d'un certain empirisme, le critère essentiel étant de produire des mouvements silencieux, avec des départs et des arrêts sans heurt. Lors des premiers essais, des rampes d'accélérations linéaires provoquaient des à coups au démarrage et des sauts de pas à l'arrêt. Ceci avait pour conséquence, lors de mouvements en cycle fermé, de ne pas remettre le robot à sa position initiale. D'autres essais ont été réalisés avec des temporisations plus courtes entre deux pas, ce qui provoquait des sautes de pas. Ces sautes de pas provenaient d'une baisse du couple moteur et se manifestaient surtout quand le bras soulevait une charge. Listing:

```
VARIABLE DELAI   20 DELAI !

: ACCELERATION  DELAI @  2 / DELAI ! ;

: DECELERATION  DELAI @  2 * DELAI ! ;

VARIABLE LIMITE

: ALLERS  ( n ——)
DUP 5 - LIMITE ! 0 SWAP
DO
   I          5 < IF DECELERATION THEN
   I LIMITE @  > IF ACCELERATION THEN
   I ACTION
LOOP ;

: RETOURS ( n ——)
DUP 5 - LIMITE ! 0 SWAP
DO
   I          5 < IF DECELERATION THEN
   I LIMITE @  > IF ACCELERATION THEN
   I ACTION
-1 +LOOP ;
```

Le programme de commande d'un organe peut être simplifié. Ainsi, en définissant:

```
: D DOIGT    SELECTION ;
: M MAIN     SELECTION ;
```

suite page 20

Listings complétant l'article du mois précédent: structures de données PL/I en FORTH.

LISTING en FIG Forth:

```
0 VARIABLE TLEN
0 VARIABLE CLEN
0 VARIABLE FLEN
( INITIALISE STRUCTURES VARIABLES)
: INIT.SV  TLEN  !  0 CLEN ! ;

( STORE STRUCTURES VARIABLES)
: STR.SV CLEN @ FLEN @ - , FLEN @ , ;

( ALLOCATE STATIC STRUCTURE)
: SS ( SIZE ---)
CREATE  DUP ALLOT INIT.SV  DOES>  ;

(BEGIN FIELD DEFINITION FOR DYNAMIC STRUCTURES)
: DS  ( SIZE ---)  INIT.SV  ;

( ALLOCATE AN ARRAY WITH A GIVEN ELEMENT SIZE)
: S.ARRAY  CREATE , ALLOT  DOES>
DUP 2+  >R @ * R> + ;

( MOVE A WHOLE STRUCTURE)
: S.MOVE  CREATE  TLEN @ ,  DOES>
@ CMOVE ;

( CREATE A FIELD  SIZE ---
  AT.RUN TIME
  STRUCTURE-ADDRESS --- FIELD ADDRESS )
: S.FLD  CREATE  CLEN @ , DUP FLEN !  CLEN +!  DOES>
@+ ;

( MOVE FROM AN ADDRESS TO A FIELD)
: FLD.MA  CREATE  STR.SV  DOES>
DUP 2+  >R @ + R> @ CMOVE ;

( MOVE FROM A FIELD TO AN ADDRESS)
: FLD.MB  CREATE  STR.SV  DOES>
>R  SWAP  R@@ + SWAP R> 2+ @ CMOVE ;

( MOVE A FIELD BETWEEN STRUCTURES)
: FLD.MC  CREATE STR.SV DOES>
>R  R@@ + SWAP R@@ + SWAP R> 2+ @ CMOVE ;

( MOVE AN ITEM OF A FIELD LENGTH)
: FLD.MD  CREATE  FLEN @ ,  DOES>
@ CMOVE ;

(PREPARE TO DEFINE SUBFIELDS)
: SF CLEN ! ;
```

LISTING en 79-STANDARD:

```
VARIABLE TLEN
VARIABLE CLEN
VARIABLE FLEN
( INITIALISE STRUCTURES VARIABLES)
: INIT.SV  TLEN  !  0 CLEN ! ;

( STORE STRUCTURES VARIABLES)
: STR.SV CLEN @ FLEN @ - , FLEN @ , ;

( ALLOCATE STATIC STRUCTURE)
: SS ( SIZE ---)
CREATE  DUP ALLOT INIT.SV  DOES>  ;

(BEGIN FIELD DEFINITION FOR DYNAMIC STRUCTURES)
: DS  ( SIZE ---)  INIT.SV  ;

( ALLOCATE AN ARRAY WITH A GIVEN ELEMENT SIZE)
: S.ARRAY  CREATE , ALLOT  DOES>
DUP 2+  >R @ * R> + ;

( MOVE A WHOLE STRUCTURE)
: S.MOVE  CREATE  TLEN @ ,  DOES>
@ CMOVE ;

( CREATE A FIELD  SIZE ---
  AT.RUN TIME
  STRUCTURE-ADDRESS --- FIELD ADDRESS )
: S.FLD  CREATE  CLEN @ , DUP FLEN !  CLEN +!  DOES>
@+ ;

: FLD.MA  CREATE  STR.SV  DOES>
DUP 2+  >R @ + R> @ CMOVE ;

( MOVE FROM A FIELD TO AN ADDRESS)
: FLD.MB  CREATE  STR.SV  DOES>
>R  SWAP  R@@ + SWAP R> 2+ @ CMOVE ;

( MOVE A FIELD BETWEEN STRUCTURES)
: FLD.MC  CREATE STR.SV DOES>
>R  R@@ + SWAP R@@ + SWAP R> 2+ @ CMOVE ;

( MOVE AN ITEM OF A FIELD LENGTH)
: FLD.MD  CREATE  FLEN @ ,  DOES>
@ CMOVE ;

(PREPARE TO DEFINE SUBFIELDS)
: SF CLEN ! ;
```

suite de la page 19

```
: P POIGNET  SELECTION ;
: C COUDE    SELECTION ;
: E EPAULE   SELECTION ;
: B BASE     SELECTION ;

: AL ALLERS ;
: RE RETOURS ;
```

Un cycle de commandes diverses peut donc être exprimé de la manière suivante:

```
B 500  AL
C 250  AL
D 1000 RE
E 300  AL
C 100  AL
B 500  RE
C 700  AL
E 800  RE
```

En reprenant ce cycle par le bas et en remplaçant tous les AL par RE et inversement, on peut faire revenir le bras à sa postition initiale.

Bibliographies:
ROBOTISEZ VOTRE THOMSON: par G.OURY ed Micro-Systèmes.
MANUEL DE REFERENCE FORTH THOMSON.

Remerciements:
Nous remercions la Sté LOISITECH pour le prêt d'un robot MULTISOFT et plus particulièrement Mr MARGUET, directeur de LOISITECH.

suite de la page 18

```
: $!  QUERY  $"  ;

: -TEXT  ( ad1 lg ad2 ---f)
2DUP + SWAP
DO DROP COUNT I C@  - DUP
  IF DUP ABS  /  LEAVE  THEN
LOOP  SWAP DROP ;

: $=  ( xxx xxx' --- f)  DUP $MAX SWAP  -TEXT ;

: .LEFT  ( xxx n ---)  TYPE SPACE  ;

: .RIGHT  ( xxx n ---)
OVER $LEN  OVER -  ROT +  SWAP TYPE SPACE ;
```